KR'98

Principles of Knowledge Representation and Reasoning

Proceedings of the Sixth International Conference

Trento, Italy
June 2–5, 1998

Edited by:
Anthony G. Cohn
Lenhart Schubert
Stuart C. Shapiro

Principles of Knowledge Representation and Reasoning:

Proceedings of the Sixth International Conference (KR '98)

Principles of Knowledge Representation and Reasoning:

Proceedings of the Sixth International Conference (KR '98)

Edited by

Anthony G. Cohn
(University of Leeds, UK)

Lenhart Schubert
(University of Rochester, New York)

Stuart C. Shapiro
(State University of New York
at Buffalo)

Morgan Kaufmann Publishers, Inc.
San Francisco, California

0905179

These proceedings were managed and produced for the organizers
of the KR '98 conference by Professional Book Center, Denver, Colorado.

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

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

ISBN 1-55860-554-1
ISSN 1046-9567

98 99 00 01 4 3 2 1

Q [illegible]

[illegible]

ENS1

CONTENTS

Preface

1998 ends the first decade of KR conferences; it also marks the 50-year anniversary of the world's first stored program digital computer (built in Manchester). Knowledge representation and automated reasoning techniques have progressed greatly in the meantime, but there remain many significant challenges for our field.

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

The traditional very high standard of papers has been maintained at KR'98. To acknowledge this, we decided to make a best paper award, based primarily on paper content but also on the presentation of the paper at the conference. (For the outcome, see the KR'98 website, http:/www.kr.org/kr.) We received 155 extended abstracts from a record 35 countries. The 55 papers we were able to include in the conference come from 19 countries.

The field of KR&R, at least as presented at this conference, seems more diverse than in previous years. The diversity is also represented in the record number of colocated events to be adjoining KR'98. These span the subareas (Description Logics, Nonmonotonic Reasoning, Formal Ontology, Verification and Validation), and also applications of KR (Interactive Multimedia Systems). A panel session was organized by Lin Padgham to explore the commonalities among these areas.

We were able to enlist three invited speakers for KR'98. Hector Levesque, the IJCAI-85 Computers and Thought lecturer, undertook to tell us "What Robots Can Do"; Maurizio Lenzerini chose to speak on a topic central to the KR conferences over the years, "Description Logics and Their Applications"; while Katharina Morik chose to address the very important topic of "How to Tailor Representations to Different Requirements."

Anthony G. Cohn
Program Co-Chair

Lenhart Schubert
Program Co-Chair

Stuart C. Shapiro
Conference Chair

Acknowledgments

KR'98 would not have been possible without the efforts of a great number of dedicated people.

This year we enlarged the size of the program committee, which reduced the overall reviewing load, and thus we hope improved the thoroughness of the reviewing procedure. We thank all the program committee members who provided thoughtful and detailed reviews, not only in the first stage of the reviewing process, but also in the second stage when reviews were exchanged and discussed and in the third stage when third or sometimes even fourth reviews were sought, often at very short notice. Our sincere thanks to them and to the aditional reviewers listed below who helped. The two program cochairs would also like to thank their secretaries, Sue Nemes (Leeds) and JoMarie Carpenter (Rochester), for their assistance.

Program Committee Members

James F. Allen
U Rochester, USA

Giuseppe Attardi
U Pisa, Italy

Franz Baader
RWTH Aachen, Germany

Fahiem Bacchus
U Waterloo, Canada

John Bell
Queen Mary & Westfield Coll, UK

Alexander Bochman
Center for Technological Ed., Israel

Mark Boddy
Honeywell Tech. Cent., USA

Alex Borgida
Rutgers U, USA

Craig Boutilier
UBC, Canada

Ronen Brafman
Ben Gurion U, Israel

Gerhard Brewka
U Leipzig, Germany

Tom Bylander
U Texas (San Ant.), USA

Marco Cadoli
U Roma (La Sap.), Italy

Ernest Davis
NYU, USA

James Delgrande
Simon Fraser U, Canada

Francesco Donini
U Roma (La Sap.), Italy

Didier Dubois
IRIT, France

Alan Frisch
U York, UK

Antony Galton
U Exeter, UK

Hector Geffner
U Simon Bolivar, Venezuela

Malik Ghallab
LAAS-CNRS, France

Robert Givan Jr.
Purdue U, USA

Georg Gottlob
TU Wien, Austria

Adam Grove
NEC Res. Inst., USA

Thomas Gruber
Intraspect Software, USA

Nicola Guarino
LADSEB-CNR, Italy

Peter Haddawy
U Wisc-Milwaukee, USA

Bernhard Hollunder
Interactive Objects Software, Germany

Henry Kautz
AT&T Labs—Research, USA

Jana Köhler
U Freiburg, Germany

Gerhard Lakemeyer
RWTH Aachen, Germany

Neal Lesh
U Rochester and MERL, USA

Vladimir Lifschitz
U Texas, USA

Gérard Ligozat
LIMSI, U Paris, France

Fangzhen Lin
Hong Kong UST, Hong Kong

Robert MacGregor
USC/ISI, USA

David McAllester
AT&T Labs—Research, USA

L. Thorne McCarty
Rutgers U, USA

Deborah McGuinness
AT&T Labs—Research, USA

Jack Minker
U Maryland, USA

Leora Morgenstern
IBM TJ Watson, USA

Bernhard Nebel
U Freiburg, Germany

Wolfgang Nejdl
U Hannover, Germany

Werner Nutt
Hebrew U, Israel

Hans Jürgen Ohlbach
Imperial College, UK

Fiora Pirri
U Roma (La Sap.), Italy

Massimo Poesio
U Edinburgh, Scotland

David Poole
UBC, Canada

Teodor Przymusinski
UC Riverside, USA

Anand Rao
Mitchell Madison Group, Australia

Raymond Reiter
U Toronto, Canada

Irina Rish
UC Irvine, USA

Stuart Russell
UC Berkeley, USA

Marco Schaerf
U Roma (La Sap.), Italy

Bart Selman
Cornell U, USA

Murray Shanahan
Queen Mary & Westfield Coll, UK

Yoav Shoham
Stanford U, USA

Maria Simi
U Pisa, Italy

Aaron Sloman
U Birmingham, UK

Michael Thielscher
U Dresden, Germany

Miroslaw Truszczyński
U Kentucky, USA

Peter van Beek
U Alberta, Canada

Mary-Anne Williams
U Newcastle, Australia

Additional Reviewers

Diego Calvanese
Giuseppe De Giacomo
Michel Devy
Graeme Hirst
Nicola Leone
Paolo Liberatore
Claudio Masolo
Fabio Massacci
David Morley
Gerald Pfeifer
Jochen Renz
Riccardo Rosati
Francesco Scarcello
Wolfgang Slany
Markus Stumptner
Hudson Turner
Helmut Veith
Franz Wotawa

We are also indebted to the very energetic and excellent local organising team, led by Fausto Giunchiglia, ably assisted by Morena Carli, Francesco Donini (Treasurer), Enrico Franconi (publicity chair), and Luciano Serafini. Theirs is an often unglamorous job, in which it is easy to attract criticism but hard to receive praise—our sincere thanks to them.

KR'98 has a record number of co-located workshops and conferences and we thank Lin Padgham (workshop coordination chair) and Paolo Traverso (local arrangements) for their efforts in facilitating this welcome addition to the KR'98 "experience." Our thanks are also due to Lin for organizing the panel session at the conference and to the panelists from the colocated events.

We also thank our sponsors and cosponsors: ITC-IRST, Fondazione CARITRO, COMULOG Net, AI*IA, and AAAI. We also thank Jennifer Ballentine and the staff of Professional Book Center for their help in producing these proceedings.

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

Building, Merging, and Revising Theories I

Description Logic Framework for Information Integration

Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, Riccardo Rosati
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italy
{calvanese,degiacomo,lenzerini,nardi,rosati}@dis.uniroma1.it

Abstract

Information Integration is one of the core problems in distributed databases, cooperative information systems, and data warehousing, which are key areas in the software development industry. Two critical factors for the design and maintenance of applications requiring Information Integration are conceptual modeling of the domain, and reasoning support over the conceptual representation. We demonstrate that Knowledge Representation and Reasoning techniques can play an important role for both of these factors, by proposing a Description Logic based framework for Information Integration. We show that the development of successful Information Integration solutions requires not only to resort to very expressive Description Logics, but also to significantly extend them. We present a novel approach to conceptual modeling for Information Integration, which allows for suitably modeling the global concepts of the application, the individual information sources, and the constraints among different sources. Moreover, we devise inference procedures for the fundamental reasoning services, namely relation and concept subsumption, and query containment. Finally, we present a methodological framework for Information Integration, which can be applied in several contexts, and highlights the role of reasoning services within the design process.

1 INTRODUCTION

In recent years there has been a growing interest in Information Integration, whose goal is to access, relate and combine data from multiple sources. Indeed, Information Integration is one of the core problems in distributed databases, cooperative information systems, and data warehousing, which are key areas in the software development industry (Wiederhold, 1996; Knoblock & Levy, 1995; Widom, 1995; Hull, 1997).

Early work on integration was carried out in the context of database design, and focused on the so-called *schema integration* problem, i.e. designing a global, unified schema for a database application starting from several subschemata, each one produced independently from the others (Batini, Lenzerini, & Navathe, 1986). More recent efforts have been devoted to *data integration*, which generalizes schema integration by taking into account actual data in the integration process. Here the input is a collection of source data sets (each one constituted by a schema and actual data), and the goal is to provide an integrated and reconciled view of the data residing at the sources, without interfering with their autonomy (Ullman, 1997). We only deal with the so-called read-only integration, which means that such a reconciled view is used for answering queries, and not for updating information.

Data integration can be either *virtual* or *materialized*. In the first case, the integration system acts as an interface between the user and the sources (Sheth & Larson, 1991; Hurson, Bright, & Pakzad, 1994), and is typical of multidatabases, distributed databases, and more generally open systems. In virtual integration query answering is generally costly, because it requires accessing the sources. In the second case, the system maintains a replicated view of the data at the sources (Gupta & Mumick, 1995; Inmon, 1996), and is typical, for example, both in information system re-engineering and data warehousing. In materialized data integration, query answering is generally more efficient, because it does not require acessing the sources, whereas maintaining the materialized views is costly, especially when the views must be up-to-date with respect to the updates at the sources (view refreshment). In the rest of this paper, we do not deal with the problem of view refreshment.

There are two basic approaches to the data integration problem, called *procedural* and *declarative*. In the procedural approach, data are integrated in an ad-hoc manner with respect to a set of predefined information needs. In this case, the basic issue is to design suitable software modules that access the sources in order to fulfill the predefined information requirements. Several data integration (both virtual and materialized) projects, such as TSIMMIS (Chawathe, Garcia-Molina, Hammer, Ireland, Papakonstantinou, Ullman & Widom, 1994; Ullman, 1997), *Squirrel* (Zhou, Hull, & King, 1996; Hull & Zhou, 1996), and WHIPS (Hammer, Garcia-Molina, Widom, Labio, & Zhuge, 1995; Wiener, Gupta, Labio, Zhuge, Garcia-Molina, & Widom, 1996) follow this idea. They do not require an explicit notion of integrated data schema, and rely on two kinds of software components: *wrappers* that encapsulate sources, converting the underlying data objects to a common data model, and *mediators* (Wiederhold, 1992) that obtain information from one or more wrappers or other mediators, refine this information by integrating and resolving conflicts among the pieces of information from the different sources, and provide the resulting information either to the user or to other mediators. The basic idea is to have one mediator for every query pattern required by the user, and generally there is no constraint on the consistency of the results of different mediators.

In the declarative approach, the goal is to model the data at the sources by means of a suitable language, to construct a unified representation, to refer to such a representation when querying the global information system, and to derive the query answers by means of suitable mechanisms accessing the sources and/or the materialized views. This is the idea underlying systems such as *Carnot* (Collet, Huhns, & Shen, 1991; Huhns, Jacobs, Ksiezyk, Shen, Singh, & Cannata, 1993), SIMS (Arens, Chee, Hsu, & Knoblock, 1993; Arens, Knoblock, & Chen, 1996) and *Information Manifold* (Levy, Srivastava, & Kirk, 1995; Kirk, Levy, Sagiv, & Srivastava, 1995; Levy, Rajaraman, & Ordille, 1996). The declarative approach provides a crucial advantage over the procedural one: although building a unified representation may be costly, it allows maintaining a consistent global view of the information sources, which represents a reusable component of the Information Integration systems.

We adopt a declarative approach to integration, and argue that two critical factors for the design and maintenance of applications requiring Information Integration are the *conceptual modeling of the domain*, and the possibility of *reasoning over the conceptual representation*. We demonstrate that Knowledge Representation and Reasoning techniques can play an important role for both of these factors, by proposing a Description Logic (Borgida, 1995; Donini, Lenzerini, Nardi, & Schaerf, 1996) based framework for Information Integration. In particular, our work provides the following main contributions:

(1) We use Description Logics for the *conceptual modeling* of both the global domain and the various sources. Since the development of successful Information Integration solutions requires specific modeling features, we propose a new Description Logic, which treats n-ary relations as first-class citizens. Note that the usual characteristic of many Description Logics to model only unary predicates (concepts) and binary predicates (roles) would represent an intolerable limit in our case.

(2) We provide suitable mechanisms for expressing what we call the *intermodel assertions*, i.e. interrelationships between concepts in different sources. Thus, integration is seen as the incremental process of understanding and representing the relationships between data in the sources, rather than simply producing a unified data schema. The fact that our approach is incremental is also important in amortizing the cost of integration.

(3) For an accurate description of the information sources, we incorporate in our logic the possibility of describing the data at the sources in terms of a set of *relational structures*. Each relational structure is defined as a view over the conceptual representation, thus providing a formal mapping between the description of data and the conceptual representation of the domain.

(4) Our representation framework is equipped with *inference procedures* for the fundamental reasoning services, namely concept and relation subsumption, and query containment. Indeed, we make use of the first decidability result on query containment for a Description Logic with n-ary relations (Calvanese, De Giacomo, & Lenzerini, 1998). Based on these reasoning methods, we present a methodological framework for Information Integration, which can be applied both in the virtual and in the materialized approach.

In comparing our framework with other declarative approaches, we observe that in both Carnot and SIMS, reasoning is based on formalisms, Cyc (Lenat

& Guha, 1990) and LOOM (MacGregor, 1991) respectively, that are undecidable. Information Manifold uses the Classic (Patel-Schneider, McGuiness, Brachman, Resnick, & Borgida, 1991) Description Logic at the conceptual level, and extends it with conjunctive queries at the logical level. While this Description Logic is polynomially decidable, it cannot fully capture neither n-ary relationships among the various classes of data in the domain, nor the intermodel assertions, nor many interesting inferences on such assertions.

Compared with the procedural approaches, which have been designed to cope in a more flexible way with the heterogeinity and the dynamics of the sources, our methodology for incremental schema integration based on intermodel assertions combines the advantages of a conceptual representation with the necessary flexibility to deal with changes in the domain. In particular, the ability of reasoning over both the conceptual representation and the relational structures can be profitably used in designing mediators with verifiable specifications.

The paper is organized as follows. In Section 2 we describe in more detail our framework for Information Integration based on Description Logics. In Section 3 we present the particular Description Logic we use in the framework. In Section 4 we illustrate how the reasoning techniques associated with our logic are used to improve the design and maintenance of the Information Integration system. Finally, Section 5 concludes the paper.

2 THE FRAMEWORK

In our approach to Information Integration, we refer to the architecture depicted in Figure 1, in which three layers can be identified:

- a *conceptual layer* called the Domain Model, which is constituted by an Enterprise Model and one Source Model for each data source;
- a *logical layer*[1], constituted by the *Source Schemas* and the *Materialized View Schema*, which describe the logical content of source data stores and of materialized view store, respectively;
- a *physical layer*, which consists of the data stores containing the actual data of the sources and the integrated materialized views.

[1]Here the term "logical" is used according to the database terminology, where it denotes a description of data in terms of structures managed by DBMSs (e.g., relational tables), which are at a more abstract level with respect to the physical organization of data.

The methodology for Information Integration described in Section 4, and the reasoning techniques illustrated in Section 3, support the incremental building of the conceptual and the logical representations. The designer is provided with information on various aspects, including the global concepts relevant for new information requirements, the sources from which a new view can be defined, the correspondences between sources and/or views, and a trace of the integration steps.

We describe now the structure of the conceptual and logical layers, which constitute the core of the proposed integration framework. The actual formalisms we adopt, and the associated reasoning techniques are described in the next section.

2.1 THE CONCEPTUAL LEVEL

The *Enterprise Model* is a conceptual representation of the global concepts and relationships that are of interest to the application. It corresponds roughly to the notion of integrated conceptual schema in the traditional approaches to schema integration. However, since we propose an incremental approach to integration, the Enterprise Model is not necessarily a complete representation of all the data of the sources but it provides a consolidated and reconciled description of the concepts and the relationships that are important to the enterprise, and have already been analyzed. Such a description is subject to changes and additions as the analysis of the information sources proceeds. The *Source Model* of an information source is a conceptual representation of the data residing in it, or at least of the portion of data currently taken into account. Again, our approach does not require a source to be fully analyzed and conceptualized.

Both the Enterprise Model and the Source Models are expressed by means of a logic-based formalism (see Section 3) which is general and powerful enough to express the usual database models, such as the Entity-Relationship Model, the Relational Model, or the Object-Oriented Data Model (for the static part). The inference techniques associated with the formalism allow for carrying out several reasoning services on the representation.

Besides the Enterprise Model and the various Source Models, the *Domain Model* contains the specification of the interdependencies between elements of different Source Models and between Source Models and the Enterprise Model. The notion of interdependency is a central one in our approach. Since the sources are of interest in the overall architecture, integration does not simply mean producing the Enterprise Model, but

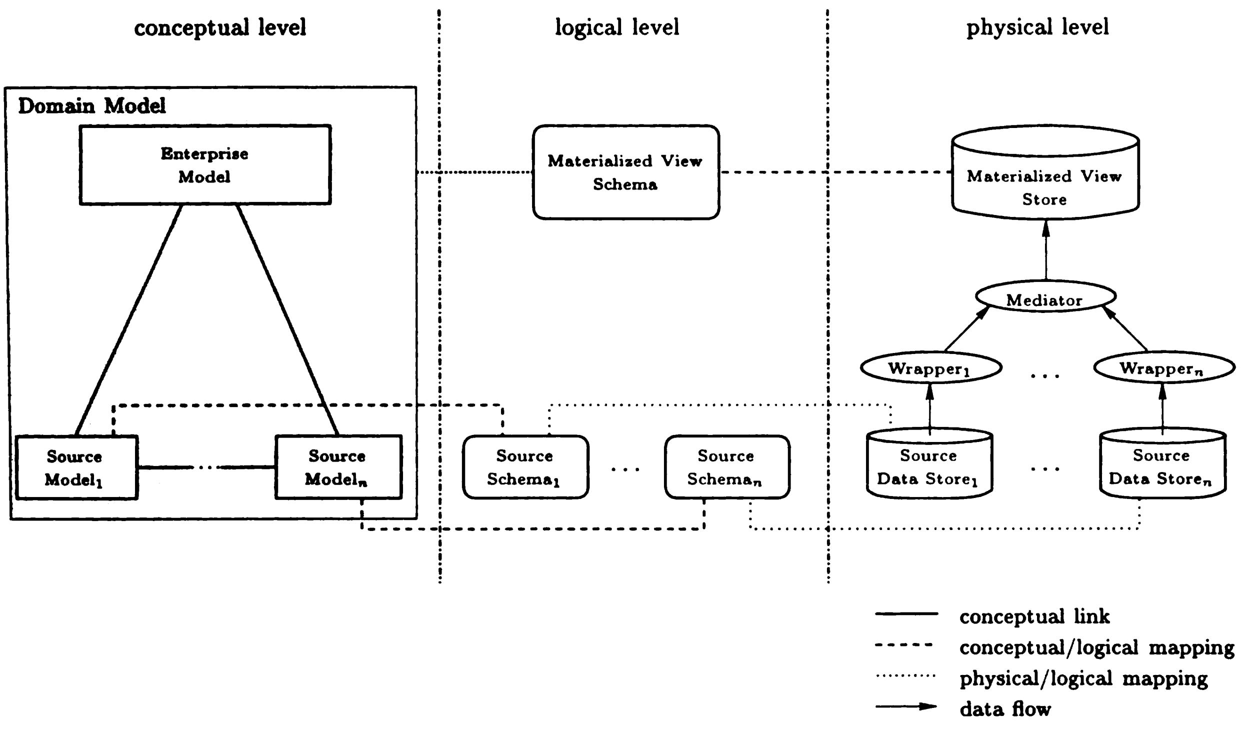


Figure 1: Architecture for Data Integration

rather to be able to establish the correct relationships both between the Source Models and the Enterprise Model, and between the various Source Models. We formalize the notion of interdependency by means of so called *intermodel assertions* (Catarci & Lenzerini, 1993), which provide a simple and effective declarative mechanism to express the dependencies that hold between entities (i.e. classes and relationships) in different models (Hull, 1997). We use again a logic-based formalism to express intermodel assertions, and the associated inference techniques provide a means to reason about interdependencies among models.

2.2 THE LOGICAL LEVEL

Our approach requires that each source, besides being conceptualized, is also described in the *Source Schema* in terms of a logical data model (in our case the Relational Model) which allows for representing the structure of the stored data. Such a structure is specified in terms of a set of relation definitions, each one expressed by means of a view (i.e. a query) over the conceptual representation of the source (i.e. the Source Model). Suitable software components, called *wrappers*, implement the mapping of physical structures to logical structures (see Figure 1). More precisey, a wrapper is able to access a source and transform the data therein into a form that is coherent with the logical specification of the source.

In the case where the integrated data (or portions thereof) are materialized, the *Materialized View Schema* provides a description of the logical content of the materialized views constituting the *Materialized View Store*. Similarly to the case of the sources, each portion of the Materialized Views Schema is described in terms of a set of definitions of relations, each one expressed in terms of a query over the Domain Model. A view is actually materialized starting from the data produced by wrappers by means of suitable software components, called *mediators* (see Figure 1). Again, a discussion on mediators is outside the scope of the present paper. In the case where a virtual approach is adopted there are no Materialized Views, and the data are provided by the mediators at query processing time.

A more detailed discussion on wrappers and mediators is outside the scope of this paper.

3 REPRESENTATION AND REASONING

In this section we present the formalism that we use for describing data both at the conceptual and the logical level, and we illustrate the basis of the reasoning techniques associated with the formalism.

3.1 REPRESENTATION AT THE CONCEPTUAL LEVEL

We use for the conceptual level a specific *Description Logic*, called $\mathcal{DLR}$, which includes *concepts* and *n-ary relations*[2]. $\mathcal{DLR}$ is inspired by the languages introduced in (Calvanese, De Giacomo, & Lenzerini, 1995; De Giacomo & Lenzerini, 1995, 1994; Catarci & Lenzerini, 1993), and is a natural extension of Description Logics (Donini et al., 1996; Calvanese, Lenzerini, & Nardi, 1994; Borgida, 1995) towards n-ary relations, which are extremely important in our context.

We assume to deal with a finite set of *atomic relations* and *concepts*, denoted by $\mathbf{P}$ and A respectively. We use $\mathbf{R}$ to denote arbitrary *relations* (of given arity between 2 and n_{max}), and C to denote arbitrary *concepts*. Concepts and relations are built according to the following syntax, where i and j denote components of relations, i.e. integers between 1 and n_{max}, n denotes the arity of a relation, i.e. an integer between 2 and n_{max}, and k denotes a nonnegative integer:

$$\begin{array}{lcl}\mathbf{R} & ::= & \top_n \mid \mathbf{P} \mid (\$i/n{:}\,C) \mid \neg\mathbf{R} \mid \mathbf{R}_1 \sqcap \mathbf{R}_2 \\ C & ::= & \top_1 \mid A \mid \neg C \mid C_1 \sqcap C_2 \mid \exists[\$i]\mathbf{R} \mid (\leq k\,[\$i]\mathbf{R})\end{array}$$

Concepts and relations must be *well-typed*, which means that (i) only relations of the same arity n can be combined to form expressions of type $\mathbf{R}_1 \sqcap \mathbf{R}_2$ (which inherit the arity n), and (ii) $i \leq n$ whenever i denotes a component of a relation of arity n.

The semantics of the $\mathcal{DLR}$ constructs is specified through the usual notion of interpretation. An *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ is constituted by an *interpretation domain* $\Delta^\mathcal{I}$ and an *interpretation function* $\cdot^\mathcal{I}$ that assigns to each concept C a subset $C^\mathcal{I}$ of $\Delta^\mathcal{I}$, and to each relation $\mathbf{R}$ of arity n a subset $\mathbf{R}^\mathcal{I}$ of $(\Delta^\mathcal{I})^n$, such that the conditions in Figure 2 are satisfied. We observe that $\top_1$ denotes the interpretation domain, while $\top_n$, for $n > 1$, does *not* denote the n-cartesian product of the domain, but only a subset of it, that covers all relations of arity n. As a consequence, the "$\neg$" construct on relations is used to express difference of relations, rather than complement.

A $\mathcal{DLR}$ *conceptual model* $\mathcal{M}$ (i.e., either the Enterprise Model or one of the Source Models) is constituted by a finite set of *intramodel assertions*, which express knowledge on the relations and concepts in $\mathcal{M}$, and have the form

$$L \sqsubseteq L' \qquad L \not\sqsubseteq L' \qquad L \equiv L' \qquad L \not\equiv L'$$

with L, L' either two relations of the same arity or two concepts.

An interpretation $\mathcal{I}$ *satisfies* an intramodel assertion $L \sqsubseteq L'$ (resp. $L \equiv L'$) if $L^\mathcal{I} \subseteq L'^\mathcal{I}$ (resp. $L^\mathcal{I} = L'^\mathcal{I}$), and it satisfies $L \not\sqsubseteq L'$ (resp. $L \not\equiv L'$) if $\mathcal{I}$ does not satisfy $L \sqsubseteq L'$ (resp. $L \equiv L'$). An interpretation *satisfies* $\mathcal{M}$, if it satisfies all assertions in $\mathcal{M}$.

To specify knowledge on the conceptual interrelationships among the sources and/or the enterprise, we use *intermodel assertions* (Catarci & Lenzerini, 1993), which have essentially the form of intramodel assertions, although the two relations (concepts) L and L' belong to two different conceptual models $\mathcal{M}_i$, $\mathcal{M}_j$. Intermodel assertions can be either *extensional*, which express relationships between the extensions of the relations (concepts) involved, or *intensional*, which express conceptual relationships that are not necessarily reflected at the instance level. Formally, an *intermodel assertion* over two conceptual models $\mathcal{M}_i$, $\mathcal{M}_j$ $(i \neq j)$ is an assertion of one of the following forms

$$\begin{array}{lll@{\qquad}lll} L & \sqsubseteq_{ext} & L' & L & \not\sqsubseteq_{ext} & L' \\ L & \equiv_{ext} & L' & L & \not\equiv_{ext} & L' \\[1em] L & \sqsubseteq_{int} & L' & L & \not\sqsubseteq_{int} & L' \\ L & \equiv_{int} & L' & L & \not\equiv_{int} & L' \end{array}$$

in which L and L' are either two relations with compatible signatures or two concepts belonging to $\mathcal{M}_i$ and $\mathcal{M}_j$ respectively.

An interpretation $\mathcal{I}$ *satisfies* an extensional intermodel assertion $L \sqsubseteq_{ext} L'$ (resp. $L \equiv_{ext} L'$) if $L^\mathcal{I} \subseteq L'^\mathcal{I}$ (resp. $L^\mathcal{I} = L'^\mathcal{I}$), and it satisfies $L \not\sqsubseteq_{ext} L'$ (resp. $L \not\equiv_{ext} L'$) if $\mathcal{I}$ does not satisfy $L \sqsubseteq_{ext} L'$ (resp. $L \equiv_{ext} L'$). Hence, interpretation of extensional intermodel assertions is analogous to the one of intramodel assertions.

Instead, intensional intermodel assertions are interpreted by first taking the intersection of the relations (concepts) L, L' with both $\top_{n_i}$ and $\top_{n_j}$ ($\top_{1_i}$ and $\top_{1_j}$). Formally:

1. Let C, C' be concepts belonging respectively to $\mathcal{M}_i, \mathcal{M}_j$. Then, an interpretation $\mathcal{I}$ satisfies the intensional intermodel assertion $C \sqsubseteq_{int} C'$ (resp. $C \equiv_{int} C'$) if

$$C^\mathcal{I} \sqcap \top_{1_i} \sqcap \top_{1_j} \subseteq C'^\mathcal{I} \sqcap \top_{1_i} \sqcap \top_{1_j}$$

(resp. $C^\mathcal{I} \sqcap \top_{1_i} \sqcap \top_{1_j} = C'^\mathcal{I} \sqcap \top_{1_i} \sqcap \top_{1_j}$). Moreover, $\mathcal{I}$ satisfies $C \not\sqsubseteq_{int} C'$ (resp. $C \not\equiv_{int} C'$) if $\mathcal{I}$ does not satisfy $C \sqsubseteq_{int} C'$ (resp. $C \equiv_{int} C'$).

[2]Domains, i.e. sets of values such as integer, string, etc., can be easily included in $\mathcal{DLR}$.

$$
\begin{array}{rcl}
\top_n^{\mathcal{I}} & \subseteq & (\Delta^{\mathcal{I}})^n \\
\mathbf{P}^{\mathcal{I}} & \subseteq & \top_n^{\mathcal{I}} \\
(\neg\mathbf{R})^{\mathcal{I}} & = & \top_n^{\mathcal{I}} \setminus \mathbf{R}^{\mathcal{I}} \\
(\mathbf{R}_1 \sqcap \mathbf{R}_2)^{\mathcal{I}} & = & \mathbf{R}_1^{\mathcal{I}} \cap \mathbf{R}_2^{\mathcal{I}} \\
(\$i/n\colon C)^{\mathcal{I}} & = & \{(d_1,\ldots,d_n) \in \top_n^{\mathcal{I}} \mid d_i \in C^{\mathcal{I}}\} \\
\\
\top_1^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \\
A^{\mathcal{I}} & \subseteq & \Delta^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C_1 \sqcap C_2)^{\mathcal{I}} & = & C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} \\
(\exists[\$i]\mathbf{R})^{\mathcal{I}} & = & \{d \in \Delta^{\mathcal{I}} \mid \exists(d_1,\ldots,d_n) \in \mathbf{R}^{\mathcal{I}}.\, d_i = d\} \\
(\leq k\,[\$i]\mathbf{R})^{\mathcal{I}} & = & \{d \in \Delta^{\mathcal{I}} \mid |\{(d_1,\ldots,d_n) \in \mathbf{R}_1^{\mathcal{I}} \mid d_i = d\}| \leq k\}
\end{array}
$$

Figure 2: Semantic rules for $\mathcal{DLR}$ ($\mathbf{P}$, $\mathbf{R}$, $\mathbf{R}_1$, and $\mathbf{R}_2$ have arity n)

2. Let $\mathbf{R}, \mathbf{R}'$ be relations (of the same arity n) belonging respectively to $\mathcal{M}_i, \mathcal{M}_j$. Then, an interpretation $\mathcal{I}$ satisfies the intensional intermodel assertion $\mathbf{R} \sqsubseteq_{int} \mathbf{R}'$ (resp. $\mathbf{R} \equiv_{int} \mathbf{R}'$) if

$$\mathbf{R}^{\mathcal{I}} \cap \top_{ni} \cap \top_{nj} \subseteq \mathbf{R}'^{\mathcal{I}} \cap \top_{ni} \cap \top_{nj}$$

(resp. $\mathbf{R}^{\mathcal{I}} \cap \top_{ni} \cap \top_{nj} = \mathbf{R}'^{\mathcal{I}} \cap \top_{ni} \cap \top_{nj}$). Moreover, $\mathcal{I}$ satisfies $\mathbf{R} \not\sqsubseteq_{int} \mathbf{R}'$ (resp. $\mathbf{R} \not\equiv_{int} \mathbf{R}'$) if $\mathcal{I}$ does not satisfy $\mathbf{R} \sqsubseteq_{int} \mathbf{R}'$ (resp. $\mathbf{R} \equiv_{int} \mathbf{R}'$).

A *Domain Model (DM)* $\mathcal{W}$ is an $(m+2)$-tuple $\langle \mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_m, \mathcal{G}\rangle$ such that: (i) $\mathcal{M}_0$ is the Enterprise Model; (ii) each $\mathcal{M}_i$, for $i \in \{1,\ldots,m\}$, is a Source Model; (iii) $\mathcal{G}$ (for "glue") is a finite set of intermodel assertions. We assume that $\mathcal{G}$ always includes for each $i \in \{1,\ldots,m\}$ the following assertions: $\top_{1i} \sqsubseteq_{ext} \top_{10}$, and $\top_{ni} \sqsubseteq_{ext} \top_{n0}$ for each n such that a relation $\mathbf{R}$ of arity n appears in $\mathcal{M}_i$. An interpretation $\mathcal{I}$ *satisfies* $\mathcal{W}$ if it satisfies all the intramodel and intermodel assertions in $\mathcal{W}$.

3.2 REPRESENTATION AT THE LOGICAL LEVEL

We express the logical level in terms of a set of relation schemas, each describing either a relation of a Source Schema, or a relation of the Materialized View Schema. Such relations are connected to the DM by characterizing each relation schema in terms of a non-recursive Datalog query over the elements of the DM, i.e. a query of the form:

$$q(\vec{\mathbf{x}}) \leftarrow body_1(\vec{\mathbf{x}}, \vec{\mathbf{y}}_1) \vee \cdots \vee body_m(\vec{\mathbf{x}}, \vec{\mathbf{y}}_m)$$

where each $body_i(\vec{\mathbf{x}}, \vec{\mathbf{y}}_i)$ is a conjunction of *atoms*, either $\mathbf{R}(\vec{\mathbf{t}})$ or $C(t)$ (where $\vec{\mathbf{t}}$ and t are variables in $\vec{\mathbf{x}}, \vec{\mathbf{y}}_i$)[3], with $\mathbf{R}$, C relations and concepts over the DM. The *arity* of q is equal to the number of variables of $\vec{\mathbf{x}}$.

[3]Our approach is applicable also when constants are used in the queries.

We observe that the atoms in the queries are arbitrary $\mathcal{DLR}$ relations and concepts, freely used in the assertions of the schema. This distinguishes our approach with respect to (Donini, Lenzerini, Nardi, & Schaerf, 1991, 1998; Levy & Rousset, 1996), where no constraints can be expressed in the schema on the relations that appear in the queries.

Given an interpretation $\mathcal{I}$ of a DM $\mathcal{W}$, a query q for $\mathcal{W}$ of arity n is interpreted as the set $q^{\mathcal{I}}$ of n-tuples $(o_1,\ldots,o_n)$, with each $o_i \in \Delta^{\mathcal{I}}$, such that, when substituting $(o_1,\ldots,o_n)$ for $(x_1,\ldots,x_n)$, the formula

$$\exists\vec{\mathbf{y}}_1.body_1(\vec{\mathbf{x}}, \vec{\mathbf{y}}_1) \vee \cdots \vee \exists\vec{\mathbf{y}}_m.body_m(\vec{\mathbf{x}}, \vec{\mathbf{y}}_m)$$

evaluates to true in $\mathcal{I}$. If q and q' are two queries (of the same arity) for $\mathcal{W}$, we say that q is *contained in* q' wrt $\mathcal{W}$, if $q^{\mathcal{I}} \subseteq q'^{\mathcal{I}}$ for every $\mathcal{I}$ satisfying $\mathcal{W}$.

3.3 REASONING

The typical kinds of reasoning services needed at the conceptual level in order to support the designer in applying the integration methodology presented in Section 4 (e.g., checking whether the DM is consistent, checking whether a relation or a concept is satisfiable in the DM, checking subsumption between relations or concepts in the DM) can be reduced to checking satisfiability of the DM. The reasoning tasks can in particular be exploited for computing and incrementally maintaining the concept and relation lattice of the DM, or more generally the lattice of all concept and relation expressions.

The expressiveness of $\mathcal{DLR}$, required for capturing meaningful properties in the DM, makes reasoning a

$$\begin{array}{rcl}
\mathtt{CONTRACT}_0 & \sqsubseteq & (\$1\colon \mathtt{Client}_0) \sqcap (\$2\colon \mathtt{Dept}_0) \sqcap \\
 & & (\$3\colon \mathtt{Service}_0) \\
\mathtt{REG\text{-}AT}_0 & \sqsubseteq & (\$1\colon \mathtt{Client}_0) \sqcap (\$2\colon \mathtt{Dept}_0) \\
\mathtt{PrDept}_0 & \sqsubseteq & \mathtt{Dept}_0 \\
 & & \\
\mathtt{REG\text{-}AT}_1 & \sqsubseteq & (\$1\colon \mathtt{Client}_1) \sqcap (\$2\colon \mathtt{Dept}_1) \\
\mathtt{PROMOTION}_1 & \sqsubseteq & \mathtt{REG\text{-}AT}_1 \\
\mathtt{LOCATION}_1 & \sqsubseteq & (\$1\colon \mathtt{Dept}_1) \sqcap (\$2\colon \mathtt{String}) \\
\mathtt{Dept}_1 & \sqsubseteq & \exists^{\leq 1}\mathtt{LOCATION}_1[\$1].\top_2 \\
 & & \\
\mathtt{CONTRACT}_2 & \sqsubseteq & (\$1\colon \mathtt{Client}_2) \sqcap (\$2\colon \mathtt{Dept}_2) \sqcap \\
 & & (\$3\colon \mathtt{Service}_2)
\end{array}$$

$$\begin{array}{rcl}
\mathtt{Dept}_1 & \equiv_{ext} & \mathtt{PrDept}_0 \\
\mathtt{REG\text{-}AT}_1 & \sqsubseteq_{ext} & \mathtt{REG\text{-}AT}_0 \\
\mathtt{Client}_1 & \equiv_{ext} & \mathtt{Client}_0 \sqcap \exists^{\geq 1}\mathtt{REG\text{-}AT}_0[\$1].\mathtt{PrDept}_0 \\
\multicolumn{3}{l}{\mathtt{Client}_0 \sqcap \exists^{\geq 1}\mathtt{CONTRACT}_0[\$1].\top_2} \\
 & \sqsubseteq_{ext} & \exists^{\geq 1}\mathtt{PROMOTION}_1[\$1].\top_2 \\
 & & \\
\mathtt{Client}_2 & \sqsubseteq_{ext} & \mathtt{Client}_0 \sqcap \exists^{\geq 1}\mathtt{CONTRACT}_0[\$1].\top_2 \\
\mathtt{Dept}_2 & \sqsubseteq_{ext} & \mathtt{Dept}_0 \\
\mathtt{Service}_2 & \equiv_{ext} & \mathtt{Service}_0 \\
 & & \\
\mathtt{Client}_1 & \equiv_{int} & \mathtt{Client}_2 \\
\mathtt{Dept}_1 & \equiv_{int} & \mathtt{Dept}_2
\end{array}$$

Figure 3: Domain model ($(\$i/n\colon C)$ is abbreviated by $(\$i\colon C)$)

complex task. We have devised a sound and complete procedure to decide the satisfiability of a DM which works in worst-case deterministic exponential time in the size of the DM. Indeed, this worst-case complexity is inherent to the problem, therefore reasoning with respect to a DM is EXPTIME complete. The inference method works in two steps: first, reasoning on the DM is reduced to reasoning on a knowledge base expressed in the Description Logic $\mathcal{CIQ}$ (De Giacomo & Lenzerini, 1996); then reasoning procedures for $\mathcal{CIQ}$, based on the correspondence with Propositional Dynamic Logics, are exploited.

For reasoning at the logical level, we provide suitable techniques for query containment. In particular, we have developed an algorithm for deciding query containment with respect to a DM, which exploits a reduction to unsatisfiability in $\mathcal{CIQ}$, and which extends the one in (Calvanese, De Giacomo, & Lenzerini, 1997; Calvanese et al., 1998) to deal with both intramodel and intermodel assertions.

3.4 EXAMPLE

Figure 3 shows a DM, $\mathcal{W} = (\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, \mathcal{G})$, that represents an enterprise and two sources containing information about contracts between clients and departments for services, and about registration of clients at departments. Symbols subscripted by i refer to model $\mathcal{M}_i$. The intramodel assertions in $\mathcal{M}_0$, $\mathcal{M}_1$, $\mathcal{M}_2$ are visualized in Figure 4, using Entity-Relationship diagrams, which are typical of conceptual modeling in Databases and are fully compatible with $\mathcal{DLR}$. Source 1 contains information about clients registered at public-relations departments. Source 2 contains information about contracts and complete information about services. The Enterprise Model provides a reconciled conceptual description of the two sources. Note that, in this example, such reconciled description is not complete yet: e.g., the relation `PROMOTION` is not modeled in $\mathcal{M}_0$ (recall that our approach to integration is incremental). The various interdependencies among relations and concepts in the Enterprise Model and the two Sources Models are represented by the intermodel assertions on the right-hand side of Figure 3.

As for the logical level representation, suppose, for example, that the actual data in Source 1 are described by a relational table $\mathtt{Table}_1$ having three columns, one for the client, one for the department which the client is registered at, and one for the location of the department. Such a table is specified in terms of the DM by means of the query:

$$\mathtt{Table}_1(x,y,z) \leftarrow \mathtt{REG\text{-}AT}_1(x,y) \wedge \mathtt{LOCATION}_1(y,z)$$

Using the reasoning services associated with $\mathcal{DLR}$, we can automatically derive logical consequences of the DM. For instance, we can prove that the assertion $\mathtt{PROMOTION}_1 \sqsubseteq_{ext} \mathtt{REG\text{-}AT}_0 \sqcap (\$2\colon \mathtt{PrDept}_0)$ is a logical consequence of $\mathcal{W}$. Observe that, although $\mathcal{M}_0$ does not contain a relation `PROMOTION`, the above assertion relates $\mathtt{PROMOTION}_1$ to $\mathcal{M}_0$ in a precise way.

Next, consider, for instance, the following queries posed to $\mathcal{M}_0$:

$$\begin{array}{rcl}
q_1(x,y) & \leftarrow & \mathtt{Client}_0(x) \wedge \mathtt{CONTRACT}_0(x,y,z) \\
q_2(x,y) & \leftarrow & \mathtt{Client}_0(x) \wedge \mathtt{CONTRACT}_0(x,y,z) \wedge \\
 & & \mathtt{REG\text{-}AT}_0(x,w) \wedge \mathtt{PrDept}_0(w)
\end{array}$$

q_2 is obviously contained in q_1. However, taking into account the assertions in $\mathcal{W}$, we can also derive that q_1 is contained in q_2 wrt $\mathcal{W}$.

4 THE METHODOLOGY

We outline a methodology for Information Integration, based on the techniques previously described, which can be applied in the context of both virtual and materialized data integration. The proposed methodology

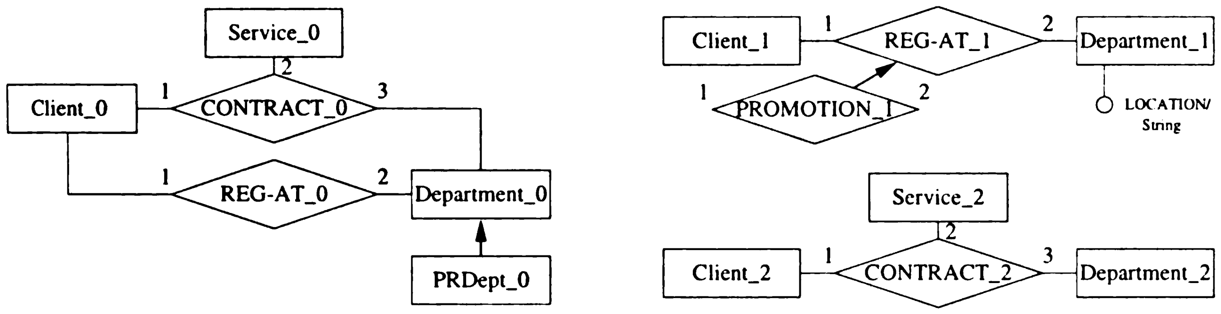


Figure 4: Enterprise and source models in Entity-Relationship diagrams

focusses on the conceptual layer of the system. Once the knowledge about this layer is available, one can exploit reasoning to support various aspects related to the other layers. We shall discuss various kinds of information that the designer can obtain through the reasoning services on the knowledge base, but the problems arising in the design of the logical and physical levels of the system are outside of the scope of the present paper. The methodology deals with two scenarios, called *source-driven* and *client-driven*. The former applies whenever the design of the system is accomplished in a top-down fashion by incrementally adding new sources of data; the latter arises when the design is developed bottom-up to satisfy the requests for data by the user applications.

4.1 SOURCE-DRIVEN INTEGRATION

Source-driven integration is triggered when a new source or a new portion of a source is taken into account for integration. The steps to be accomplished in this case are:

(1) *Source Model construction.* The Source Model capturing the concepts and the relationships of the new source that are critical for the enterprise is produced. Since in a typical setting, sources already exist, this task may be accomplished through a reverse engineering activity (Batini, Ceri, & Navathe, 1992). However, it is worth stressing that the Source Model is really meant to capture the semantics of the domain, independently of the organization of the data recorded in the physical structures. To this end, our approach provides a very expressive modeling language, which, as already pointed out, embodies the features of the most popular data models. In addition, in our formalism, it is possible to reason about the Source Model. Once a formalization of the model in terms of $\mathcal{DLR}$ is provided, checking several interesting properties of the model becomes possible, and can be used to help correctness and optimality of the design. We refer to (Calvanese et al., 1994) for a discussion on using the inference techniques associated to Description Logics during Source Model construction.

(2) *Source Model integration.* The Source Model is *integrated into the Domain Model.* This can lead to changes both to the Source Models, and to the Enterprise Model. It is worth recalling that our approach to integration is mainly declarative: the most important efforts during source integration are thus devoted to single out and to specify the intermodel assertions relating the Enterprise Model and the Source Models, rather than producing a unified conceptual representation. More precisely, the step of Source Model integration is characterized by the following activities:

- Structural and semantic conflicts involving the Source Model under analysis are detected and solved.
- Intermodel assertions between the Source Model and the Enterprise Model, and between different sources, are added to the Domain Model.

The activity of conflict resolution in our framework can be carried out by relying on the large body of work developed in database integration. More specifically, in our framework, the basic structural and semantic conflicts are very similar to those arising in the Entity-Relationship Data Model. An example of structural conflict is represented by the situation where the same concept is represented as a class in one model and as a relation in another model. The principles for resolving such conflicts are now well established (Batini et al., 1992). Other types of conflicts are dealt with in the Quality Analysis step.

The specification of intermodel assertions and the derivation of implicit relationships by exploiting the reasoning techniques, represent the novel part of the methodology. The most common intermodel assertions are those specifying the relation between elements in one Source Model with elements in the Enterprise Model. However, also assertions relating elements in different Source Models are of importance. For example, inferring that the set of instances of a relation in source S_i is always a subset of those in source S_j can be important in order to infer that accessing source S_j for retrieving instances of the relation is useless. We point out that the possibility of expressing relationships between concepts in different sources is a distinguished feature of our approach.

Intermodel assertions can be roughly classified as follows:

- *Subsetting assertions*, that are used to state that a certain concept or relation in the Source Model is a subset of another concept or relation in the Enterprise Model (or in another source). These assertion have the form $L_i \sqsubseteq_{ext} L'_j$.
- *Definition assertions*, that are used to completely characterize the set of instances of one concept in a model in terms of the set of instances of a concept in another model.
- *Completeness assertions*, that are used to state that the set of instances of a concept or relation in the Enterprise Model can be obtained as the union of different concepts or relations in the various sources. A special case of this type of assertions is the one stating that a certain concept in the Enterprise Model is fully captured by a concept in one Source Model.
- *Synonym assertions*, that are used to state that different symbols in two models denote in fact the same concept. These assertions have the form $L_i \equiv_{int} L_j$.
- *Homonym assertions*, that are used to state that the same symbol is used to denote different concepts in different models. These assertions have the form $L_i \not\equiv_{int} L_j$.

It is important to observe that the possibility of using complex concept and relation expressions in the context of intermodel assertions greatly enhances the expressive power of such assertions, and is another distinguished feature of our approach. Moreover, the possibility of reasoning about intermodel assertions provides support and guidelines to the designer of the Domain Model, as pointed out in the discussion on the step of quality analysis. Finally, we note that the usage of intermodel assertions is required also to reason about queries which is addressed in client-driven integration.

(3) *Quality analysis.* The goal of this step is to verify that the quality requirements are met by the Domain Model. In particular, the reasoning capabilities of our approach allow for dealing with several quality factors, such as:

- *Consistency* of the Source Model in isolation.
- *Redundancy*, by identifying equivalent concepts.
- *Readability*, by pointing out relationships that are implicit in the model.
- *Accessibility*, which amounts to verifying which data are available in the Materialized View Store, which data can be extracted from the sources, and which are indeed needed from external sources.
- *Believability*, which amounts to verifying whether the data available in the materialized views or provided by a source are consistent and complete.

It is worth noticing that, depending on the result of the evaluation of the quality factors, a restructuring of both the Source Model and the Enterprise Model may be required.

(4) *Source Schema specification.* The Source Schema, i.e. the logical view of the new source or a new portion of the source (expressed as a collection of queries over the corresponding Source Model) is specified. The source schemas are used in order to determine the sources relevant for computing answers to queries, by exploiting the ability to reason about queries. Notice that, the actual logical design of the sources is outside the scope of the integration system. Therefore, the focus here is on the specification of the sources at the logical level.

(5) *Materialized View Schema restructuring.* This step is done only in Materialized Data Integration. As we said before, the Materialized View Schema is specified in terms of a set of relational tables, each one described as a query over the Domain Model. On the basis of the description of the new source, an analysis can be carried out on whether the Materialized View Schema should be restructured and/or modified in order to better satisfy quality criteria. Again several quality factors can be evaluated by exploiting reasoning, which, in this case, essentially amounts to query containment. Although the design of the Materialized View Schema is outside the scope of the present work, we point out that this task can be effectively supported by the reasoning services about the representation of the logical schemata in terms of queries.

4.2 CLIENT-DRIVEN INTEGRATION

The client-driven design strategy refers to the case when a new query (or a set of queries) posed by a client is considered. The query is expressed in terms of the domain model, and the reasoning facilities are exploited to analyze and systematically decompose the query and check whether its components are subsumed by the views defined in the various schemas. Therefore, the central reasoning service for query analysis is query containment checking.

In Materialized Data Integration, the analysis is carried out as follows:

(1) We verify whether and how the answer can be computed from the materialized views. This problem is known as the *query rewriting* problem, which amounts to find a way to rewrite the original query in terms of the relations in the Materialized View Schema. Although we do not have a method for automatically rewriting the query, we can exploit query containment checking in order to support the designer in this task.

(2) In the case where the materialized views are not sufficient to compute the answer to a query, the idea is to verify whether the answer can be obtained by materializing new concepts represented in the Domain Model. It is interesting to observe that this is again an instance of the query rewritinng problem, where one aims at expressing the query in terms of the relations in the Sources. In this case, query containment helps to identify the set of subqueries to be issued on the sources and to extend and/or restructure the Materialized View Schema (see step 5 of source-driven integration). Different choices can be identified, based on various preference criteria. E.g., in (Levy et al., 1995) minimization in terms of the number of sources is proposed, based on the observation that accessing a source is the most expensive part of the process. In fact, by exploiting the information available through the intermodel assertions, we can accommodate different kinds of constraints, that are related to the above mentioned quality factors. For example, we can optimize with respect to believability, or interpretability, possibly combining different factors.

(3) In the case where neither the materialized data nor the concepts in the Domain Model are sufficient, the necessary data should be searched for in new sources, or in new portions of already analyzed sources. The new (portions of the) sources are then added to the Domain Model using the source-driven approach, and the process of analyzing the query is iterated.

In Virtual Data Integration, the basic problem is to determine whether and how the answer can be computed from the data in the analyzed sources, falling into case (2) or (3).

5 CONCLUSIONS

In this paper we have presented the fundamental features of a declarative approach to Information Integration based on Description Logics. As pointed out in the previous sections, there are a number of issues that deserve further investigation, and in particular:

- How to exploit the knowledge about the conceptual level in the design of wrappers and mediators.
- Designing automatic methods and techniques for the query rewriting problem, arising in the client-driven integration.

We are currently studying the above issues within the ESPRIT Project DWQ (Foundations of Data Warehouse Quality) (Calvanese, De Giacomo, Lenzerini, Nardi, & Rosati, 1997), where we are using the presented framework in the context of data warehouse design.

Acknowledgments

This work was partly supported by MURST, ESPRIT LTR Project No. 22469 DWQ (Foundations of Data Warehouse Quality), the Italian Research Council (CNR) under Progetto Strategico "Informatica nella Pubblica Amministrazione", sottoprogetto PROGRESS (Reingegnerizzazione dei Processi e dei Dati nella Pubblica Amministrazione), and the Italian Space Agency (ASI) under project "Integrazione ed Accesso a Basi di Dati Eterogenee".

References

Arens, Y., Chee, C. Y., Hsu, C., & Knoblock, C. A. (1993). Retrieving and integrating data from multiple information sources. *Journal of Intelligent and Cooperative Information Systems*, *2*(2), 127–158.

Arens, Y., Knoblock, C. A., & Chen, W. (1996). Query reformulation for dynamic information integration. *Journal of Intelligent Information Systems*, *6*, 99–130.

Batini, C., Ceri, S., & Navathe, S. B. (1992). *Conceptual Database Design, an Entity-Relationship Approach*. Benjamin and Cummings Publ. Co., Menlo Park, California.

Batini, C., Lenzerini, M., & Navathe, S. B. (1986). A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, *18*(4), 323–364.

Borgida, A. (1995). Description logics in data management. *IEEE Transactions on Knowledge and Data Engineering*, 7(5), 671–682.

Calvanese, D., De Giacomo, G., & Lenzerini, M. (1995). Structured objects: Modeling and reasoning. In *Proc. of the 4th Int. Conf. on Deductive and Object-Oriented Databases (DOOD-95)*, No. 1013 in Lecture Notes in Computer Science, pp. 229–246. Springer-Verlag.

Calvanese, D., De Giacomo, G., & Lenzerini, M. (1997). Conjunctive query containment in Description Logics with n-ary relations. In *Proc. of the 1997 Description Logic Workshop (DL-97)*, pp. 5–9.

Calvanese, D., De Giacomo, G., & Lenzerini, M. (1998). On the decidability of query containment under constraints. In *Proceedings of the Seventeenth ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS-98)*. To appear.

Calvanese, D., De Giacomo, G., Lenzerini, M., Nardi, D., & Rosati, R. (1997). Source integration in data warehousing. Tech. rep. DWQ-UNIROMA-002, DWQ Consortium.

Calvanese, D., Lenzerini, M., & Nardi, D. (1994). A unified framework for class based representation formalisms. In Doyle, J., Sandewall, E., & Torasso, P. (Eds.), *Proc. of the 4th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-94)*, pp. 109–120 Bonn. Morgan Kaufmann, Los Altos.

Catarci, T. & Lenzerini, M. (1993). Representing and using interschema knowledge in cooperative information systems. *Journal of Intelligent and Cooperative Information Systems*, *2*(4), 375–398.

Collet, C., Huhns, M. N., & Shen, W.-M. (1991). Resource integration using a large knowledge base in Carnot. *IEEE Computer*, *24*(12), 55–62.

De Giacomo, G. & Lenzerini, M. (1994). Description logics with inverse roles, functional restrictions, and n-ary relations. In *Proc. of the 4th European Workshop on Logics in Artificial Intelligence (JELIA-94)*, Vol. 838 of *Lecture Notes in Artificial Intelligence*, pp. 332–346. Springer-Verlag.

De Giacomo, G. & Lenzerini, M. (1995). What's in an aggregate: Foundations for description logics with tuples and sets. In *Proc. of the 14th Int. Joint Conf. on Artificial Intelligence (IJCAI-95)*, pp. 801–807.

De Giacomo, G. & Lenzerini, M. (1996). TBox and ABox reasoning in expressive description logics. In Aiello, L. C., Doyle, J., & Shapiro, S. C. (Eds.), *Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-96)*, pp. 316–327. Morgan Kaufmann, Los Altos.

Donini, F. M., Lenzerini, M., Nardi, D., & Schaerf, A. (1991). A hybrid system integrating Datalog and concept languages. In *Proc. of the 2nd Conf. of the Italian Association for Artificial Intelligence (AI*IA-91)*, No. 549 in Lecture Notes in Artificial Intelligence. Springer-Verlag. An extended version appeared also in the Working Notes of the AAAI Fall Symposium "Principles of Hybrid Reasoning".

Donini, F. M., Lenzerini, M., Nardi, D., & Schaerf, A. (1996). Reasoning in description logics. In Brewka, G. (Ed.), *Principles of Knowledge Representation*, Studies in Logic, Language and Information, pp. 193–238. CSLI Publications.

Donini, F. M., Lenzerini, M., Nardi, D., & Schaerf, A. (1998). AL-log: Integrating datalog and description logics. *Journal of Intelligent Information Systems*. To appear.

Patel-Schneider, P., McGuiness, D., Brachman, R. J., Resnick, L., & Borgida, A. (1991). The CLASSIC knowledge representation system: Guiding principles and implementation rational. *SIGART Bulletin*, *2*(3), 108–113.

Chawathe, S., Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J., & Widom, J. (1994). The TSIMMIS project: Integration of heterogeneous information sources. In *Proc. of IPSI Conference (IPSI'94)*.

Gupta, A. & Mumick, I. S. (1995). Maintenance of materialized views: Problems, techniques, and applications. *IEEE Bulletin of the Technical Committee on Data Engineering*, *18*(2), 3–18.

Hammer, J., Garcia-Molina, H., Widom, J., Labio, W., & Zhuge, Y. (1995). The Stanford data warehousing project. *IEEE Bulletin of the Technical Committee on Data Engineering*, *18*(2), 41–48.

Huhns, M. N., Jacobs, N., Ksiezyk, T., Shen, W.-M., Singh, M. P., & Cannata, P. E. (1993). Integrating enterprise information models in Carnot. In *Proc. of the Int. Conf. on Cooperative Information Systems (CoopIS-93)*, pp. 32–42.

Hull, R. (1997). Managing semantic heterogeneity in databases: A theoretical perspective. In *Proc. of the*

16th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS-97).

Hull, R. & Zhou, G. (1996). A framework for supporting data integration using the materialized and virtual approaches. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pp. 481–492.

Hurson, A., Bright, M., & Pakzad, S. (Eds.). (1994). *Multidatabase Systems: An Advanced Solution for Global Information Sharing.* IEEE Computer Society Press.

Inmon, W. H. (1996). *Building the Data Warehouse* (second edition). John Wiley & Sons.

Kirk, T., Levy, A. Y., Sagiv, Y., & Srivastava, D. (1995). The Information Manifold. In *Proceedings of the AAAI 1995 Spring Symp. on Information Gathering from Heterogeneous, Distributed Enviroments*, pp. 85–91.

Knoblock, C. & Levy, A. (Eds.). (1995). *AAAI Symposium on Information Gathering from Heterogeneous, Distributed Environments*, No. SS-95-08 in AAAI Spring Symposium Series. AAAI Press/The MIT Press.

Lenat, D. & Guha, R. V. (1990). *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project.* Addison Wesley Publ. Co., Reading, Massachussetts.

Levy, A. Y., Rajaraman, A., & Ordille, J. J. (1996). Query answering algorithms for information agents. In *Proc. of the 13th Nat. Conf. on Artificial Intelligence (AAAI-96)*, pp. 40–47.

Levy, A. Y. & Rousset, M.-C. (1996). CARIN: A representation language combining Horn rules and description logics. In *Proc. of the 12th European Conf. on Artificial Intelligence (ECAI-96)*, pp. 323–327.

Levy, A. Y., Srivastava, D., & Kirk, T. (1995). Data model and query evaluation in global information systems. *Journal of Intelligent Information Systems*, *5*, 121–143.

MacGregor, R. (1991). Inside the LOOM description classifier. *SIGART Bulletin*, *2*(3), 88–92.

Sheth, A. & Larson, J. (1991). Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, *22*(3).

Ullman, J. D. (1997). Information integration using logical views. In *Proc. of the 6th Int. Conf. on Database Theory (ICDT-97)*, Lecture Notes in Computer Science, pp. 19–40. Springer-Verlag.

Widom, J. (1995). Special issue on materialized views and data warehousing. *IEEE Bulletin on Data Engineering*, *18*(2).

Wiederhold, G. (1992). Mediators in the architecture of future information systems. *IEEE Computer* (March), 38–49.

Wiederhold, G. (1996). Special issue: Intelligent integration of information. *Journal of Intelligent Information Systems*, *6*(2/3).

Wiener, J. L., Gupta, H., Labio, W. J., Zhuge, Y., Garcia-Molina, H., & Widom, J. (1996). A system prototype for warehouse view maintenance. Tech. rep., Stanford University. Available at http://www-db-stanford.edu/warehousing/warehouse.html.

Zhou, G., Hull, R., & King, R. (1996). Generating data integration mediators that use materializations. *Journal of Intelligent Information Systems*, *6*, 199–221.

A Completeness Result for Reasoning with Incomplete First-Order Knowledge Bases

Hector Levesque
Dept. of Computer Science
University of Toronto
Toronto, Ont. M5S 3H5
hector@cs.toronto.edu

Abstract

Perhaps the simplest and most common method of dealing with incomplete knowledge in AI systems involves working with a third truth value standing for "unknown." By extending the ordinary two-valued truth tables in the obvious way, what is known about complex logical formulas can be efficiently calculated as a function of what is known about the atomic ones. While this form of reasoning can be shown to be sound, it is notoriously incomplete in classical logic. In this paper, we extend this reasoning method to include quantifiers and equality, and show that efficiency and soundness are preserved. We also show that for a wide class of first-order formulas in a certain normal form, the method is also complete. Finally, we prove that in the propositional case, every formula can be converted to a logically equivalent one in this normal form, and conjecture that this remains true in the first-order case.

1 INTRODUCTION

From the very beginnings of AI, the dream of getting a machine to exhibit common sense was linked to deductive reasoning:

We shall therefore say that a program has common sense if it automatically deduces for itself a sufficiently wide class of immediate consequences of anything it is told and what it already knows.
— John McCarthy in [14]

Since then, the enthusiasm for deduction has been tempered somewhat by what has been discovered about its computational difficulty. Regardless of how one feels about the relevance of complexity and computability theory to AI, for knowledge bases (KBs) large enough to hold what is presumed to be necessary for human-level common sense, deduction would have to be *extremely* efficient. Recent local search based methods like GSAT [17] do show some promise on large KBs, but so far (1) they are restricted to constraint satisfaction tasks not deductive ones, and (2) they work only on problems that can be formulated in a propositional language.[1]

To the best of our knowledge, there is so far only one logically correct (sound and complete) deductive technique efficient enough to be feasible on KBs of this size: the deduction underlying database retrieval. In KR terms, this amounts to what was called *vivid reasoning* in [11]. In logical terms, the requirements for this form of reasoning are clear: every relevant atomic formula must be known to be true or known to be false. That is, the KB must be equivalent to a maximally consistent set of literals. In addition, this set of literals must be readily computable. In the propositional case, one obvious way of ensuring this is to store the positive ones in a database and infer the negative ones using negation as failure. With every atom known true or known false, it then follows that every formula can be "efficiently" (in a sense to be discussed later) determined to be true or to be false by *evaluating* it, that is, by calculating its truth value as a function of the truth values of its constituent atoms.

But this requirement for complete knowledge is very strict. It would certainly be desirable to allow some atomic formulas to be *unknown*, with the understanding that other formulas would need to be unknown as well. Allowing arbitrary disjunctions (or existential quantifications) in the KB would obviously require a very different method of reasoning. A less radical move, which still allows incomplete knowledge, is to consider a KB that is equivalent to a finite consistent set of literals, not necessarily maximal. Unfortunately, although this is a trivial extension to the above, we can already see that it will not work: for the special case of a KB equivalent to the empty set of literals, the formulas

[1]Converting first-order reasoning problems into propositional ones remains a possibility (as done in [8], for example), but consider that for KBs with (say) 10^5 unique names, even a single binary predicate would generate far too many atomic propositions.

that would need to be known are precisely the *valid* ones. Computing these is co-NP hard in the propositional case, and even if we accept the argument that it may still be feasible in practice (perhaps because the query will always be small, or for reasons like those discussed in [7]), there is no escaping the fact that it would be undecidable in the first-order case.

So it appears that even a seemingly insignificant increase in expressive power, allowing for the most basic form of incompleteness in the KB, already makes deduction too hard. Despite this, it is precisely this form of incomplete knowledge that we will attempt to handle in this paper, suitably generalized to deal with quantifiers and equality. What we will prove is this:

1. when an evaluation-based reasoning procedure is applied to incomplete KBs of the above sort, it remains efficient and always provides logically sound answers;
2. for a wide class of queries in a special normal form ($\mathcal{NF}$), it also provides logically complete answers (and hence is logically correct);
3. we conjecture (and prove in the propositional case) that this special normal form entails no loss of expressive power, in the sense that every query can be equivalently expressed as a formula in $\mathcal{NF}$.

We need to clear about one thing at the outset: there is no free lunch. If the evaluation-based reasoning procedure is logically correct and efficient, then *converting a formula into $\mathcal{NF}$ will be computationally intractable.* In the first-order case, if my conjecture is correct, it must be undecidable!

So the conversion to normal form is not something we would want to do online. The application we have in mind is this: assume we have a knowledge-based program (problem solver, planner, whatever) that must use a very large external KB for some task. Embedded within the program are a number of queries to the KB, that is, a number of places where the program needs to know whether or not something is true. In an offline manner, we ensure these queries are in $\mathcal{NF}$ (perhaps by hand) before even approaching the KB. Once this is done, we can run the program that uses our deductive procedure with confidence, since we know it will efficiently generate answers that are logically sound and complete.

One caveat: even if we perform this conversion to $\mathcal{NF}$ by hand, it might still be the case that the $\mathcal{NF}$ is exponentially longer than the original query. We do not see this as a major problem. In the intended application, like with databases, we expect the query to be so small compared to the size of the KB that a worst-case exponential blowup (and the work this entails) is less of a concern. We will also end up assuming that within a first-order query, the depth of nesting of quantifiers is small relative to the size of the query. Note that these assumptions need not hold in a more mathematical (or "puzzle-mode") setting. There, the query (the theorem to be proved) might be about the same size as the KB (the given axioms), and the depth of quantifiers might be significant. For our application, think something like: depth of quantifiers < 4, length of query < 18 terms, and length of KB $< 10^6$ terms.

In the next section we discuss evaluation-based reasoning in general terms. In Section 3, we proceed with the development in detail for a first-order logic with equality, and state the main results. Finally, in Section 4 we draw some conclusions and describe some future work.

2 EVALUATION-BASED REASONING

The reasoning procedure we have in mind (for KBs with both complete and incomplete knowledge) is one that decides whether a formula is true or false, by evaluating it, reducing knowledge of complex formulas to knowledge of the ground atomic formulas, $\mathcal{A}$. Throughout, we will use 0 to mean "known to be false," 1 to mean "known to be true," and $\frac{1}{2}$ to mean "unknown."[2]

Given an assignment $V \in [\mathcal{A} \rightarrow \{0, 1, \frac{1}{2}\}]$ telling us which atoms are known, we extend the assignment to all boolean formulas in the obvious way:

1. $V[\neg\alpha] = 1 - V[\alpha]$.
2. $V[\alpha \wedge \beta] = \min\{V[\alpha], V[\beta]\}$.

This is merely a compact way of expressing the 3-valued truth tables first presented by Kleene [6]:

		$p \wedge q$			$\neg p$
	q	T	U	F	
	T	T	U	F	F
p	U	U	U	F	U
	F	F	F	F	T

Disjunctions, implications, and equivalences can be handled as abbreviations. We will sometimes also use the logical constant TRUE, with $V[\text{TRUE}] = 1$.

To handle quantification, assume we are given a finite set H of constants (intuitively, those names mentioned in some KB), and we define

3. $V[\forall x.\alpha] = \min_{c \in H^+}\{V[\alpha^x_c]\}$

Here α^x_c is the result of replacing free x by c in α, and H^+ is the union of the constants in H, those mentioned in α, and one new one outside of H and not mentioned in α. Thus,

[2]If we were to allow for inconsistent KBs as well, we would have a *fourth* truth value, as in [1, 3, 4, 5, 9, 10, 16], among many others. From an efficiency point of view, nothing is gained by this move, so we forego it for simplicity.

to evaluate $\forall x.\alpha$, we evaluate a finite set of its instances where the x ranges over the constants in the given H, over the constants mentioned in α, and over one new constant that is neither in H nor in α. We handle existentials as abbreviations.

Finally to handle equality formulas, we use the simplest possible scheme (for ground atomic ones):

4. $V[t = t'] = 1$ if t is identical to t', and 0 otherwise.

So all that is left to completely determine a V function is the set H and the value of V on atomic formulas. We will show how to get these from a given KB in Section3.3. Then, using these four rules, we can evaluate any closed formula, that is, how to compute what is known about the formula as a function of what is known about instances of its atoms.

Of course it remains to be seen whether this 3-valued evaluation scheme is any good. This is what is addressed in Sections 3.4 and Sections 3.5.

We should be clear about what we mean by correctness. We will want to talk about making deductions from a set of formulas S (the KB), and getting the correct answer (0, 1, or $\frac{1}{2}$) for a class of formulas T (the potential queries):

Definition 1: Let $S, T \subseteq \mathcal{L}$, and let $f \in [\mathcal{L} \rightarrow \{0, 1, \frac{1}{2}\}]$. Then

- f is logically <u>sound</u> 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 <u>complete</u> 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 $f[\alpha] = 0$;
- f is logically <u>correct</u> wrt S for T iff it is both sound and complete.

We will see below (after we establish some properties of quantifiers and equality) that whenever we begin with an evaluation function that is logically sound for atomic formulas, it will end up logically sound for all formulas. But this will not be the case for logical completeness: it is a well known property of multi-valued logics [18] that classically correct answers for atoms do not guarantee correctness for all formulas.

Observe, for example, that we would want $V[p \vee \neg p]$ to be 1 even when $V[p] = \frac{1}{2}$, contrary to what we have above. This has suggested to some authors that perhaps tautologies and their negations need to be filtered out separately in the evaluation (as in [20] and in supervaluations [19]).

But the problem is not merely with tautologies. Suppose we have that $V[p] = \frac{1}{2}$, $V[q] = 1$ and $V[r] = 0$ (where *e.g.* $KB = \{q, \neg r\}$). Let α be the formula

$$(q \wedge (\neg r \wedge p)) \vee (\neg p \wedge (\neg r \wedge q)).$$

Then, we get $V[\alpha] = \frac{1}{2}$, whereas completeness requires a value of 1 (since $KB \models \alpha$). There is, however, a tautology hidden here: if we convert α to CNF, we get

$$[q \wedge \neg r \wedge (p \vee \neg p)],$$

which gives a value of 1, after we filter out the tautologous clause.

But consider the dual of α: $[(\neg q \vee r \vee p) \wedge (\neg q \vee r \vee \neg p)]$. For logical completeness, this should get value 0, although again V returns $\frac{1}{2}$. Moreover, the formula here is in CNF, and there are no hidden tautologous clauses to remove.[3] However, observe that the clause $(\neg q \vee r)$ is derivable from these two by Resolution, and if we were to conjoin this new clause to the formula, logical equivalence would be preserved and V would now return the correct answer, 0. This is the idea behind the normal form we will introduce later.

Notice that $(p \vee \neg p)$ (or any other valid formula) is an unusual query to appear in a knowledge-based program. We couldn't possibly want our program to do one thing when it was true, and another when it was false, for instance. The formula ought to be known, arguably, but only for logical reasons, not because of anything in the KB. Similarly, the formula α above ought to be known, but its truth is hidden in a logical puzzle. Our conversion to normal form insists on unpacking these logical puzzles within the query, and making explicit what we are asking regarding the KB. One way of saying this is that a disjunction in a query must not be entailed unless one of the disjuncts in the query is, or equivalently, the conjuncts in a query must not together entail anything that is not already entailed by one of them. This is what is behind our notion of "logically separable" below.

A few words on the efficiency of the above treatment of knowledge. If the query does not use quantifiers, V will ask for the value of an atom a linear (in the size of the query) number of times. So non-quantified queries are handled efficiently, assuming atoms are. But for quantified queries, the situation is less clear. Consider one like $\exists x_1 \cdots \exists x_n(\rho_1 \wedge \cdots \wedge \rho_m)$, where the ρ_j are atoms whose arguments are among the x_i. Even if we imagine a KB that is a simple database (a finite set of ground atoms) that uses k constants, the obvious way of handling this requires looking at all k^n vectors of constants, clearly infeasible for the sort of large k we are considering.[4] In actual database systems, queries like this can be formulated, but they are handled in practice using a number of optimizations such as sort restrictions on variables (so that not all constants need be considered for every variable), and bottom-up relational operations (like merge, join, and project). These types of optimizations will be available to us as well, and coupled with our assumption that n is very small, we take it that quanti-

[3]We could convert the formula to DNF and remove the complement of tautologous clauses, and that would work here, but not in the first-order case. See below.

[4]The worst case complexity of this problem does not look good, but is not an issue here.

fied queries can be handled efficiently (or as efficiently as can be expected), assuming again that atomic queries are.

3 FIRST-ORDER KBS AND QUERIES

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.

3.1 QUANTIFIERS AND EQUALITY

Because we will be considering KBs and queries that use equality, we will end up wanting to compute the entailments not just of the KB, but of $\mathcal{E} \cup KB$, where we have:[5]

Definition 2: The set $\mathcal{E}$ is the axioms of equality (reflexivity, symmetry, transitivity, substitution of equals for equals) and the (infinite) set of formulas $\{(c_i \neq c_j) \mid i \neq j\}$.

Note that because we are making a unique-name assumption for infinitely many constants, we will not be able to finitely "propositionalize" first-order KBs, despite the lack of function symbols. 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 let ρ range over atoms whose arguments are distinct variables, so that $\rho\theta$ ranges over ground atoms. We will use $\forall\alpha$ to mean the universal closure of α. When S is finite, $\wedge S$ stands for the conjunction of its elements (and the logical constant TRUE, when S is empty). Finally, we will use e to range over *ewffs*, by which we mean quantifier-free formulas whose only predicate is equality.

Before discussing KBs and queries, we need to establish how the quantifiers and substitution behave. First we define the notion of a standard interpretation:

Definition 3: A <u>standard</u> 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.

The following is easy to show:

Lemma 1: *A standard interpretation M has the following properties:*

- $M \models \mathcal{E}$;
- $M \models e\theta$ *iff* $\mathcal{E} \models e\theta$;
- $M \models \forall x.\alpha$ *iff for every* $c \in \mathcal{C}$, $M \models \alpha^x_c$.

[5]To make the presentation here general and self-contained, we use ordinary first-order logic, with additional axioms for equality. But perhaps a better approach is to use a logical language like that presented in [9, 13] where equality is a special primitive, and the language includes "standard names" (in addition to constants and function symbols), for which the unique-name assumption is inherent in the semantics.

We get the following theorem:

Theorem 2: *Suppose S is any set of closed wffs, 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.*

Proof: Suppose we are given a model M of $\mathcal{E} \cup S$. We will show how to construct a model M' that is standard. We assume without loss of generality that the domain of M is countable, and because of $\mathcal{E}$, infinite. We begin by partitioning this domain into equivalence classes relative to the interpretation of = in M. The domain of M' will be these classes, the interpretation of = in M' will be the identity relation, and for any predicate Q, the interpretation of Q in M' is formed by taking the interpretation of Q in M and moving to the corresponding tuples of equivalence classes. Finally, to interpret a constant in M', we take the equivalence class of its denotation when the constant appears in S; otherwise, for the remaining countably infinite set of constants, we assign them in M' systematically to the remaining countably infinite set of equivalence classes, in such a way that each class is denoted by some constant. Then, it only remains to be shown that $M' \models S$. More generally, we can show that if μ is a mapping from variables to the domain of M, and μ' is the corresponding assignment to equivalence classes for M', then for any formula α all of whose constants appear in S, we have that $M, \mu \models \alpha$ iff $M', \mu' \models \alpha$. This is done by induction on the length of α. ∎

This is like Herbrand's Theorem (with $\mathcal{C}$ being like the Herbrand Universe) except that S is not required to be in prenex form, can contain arbitrary alternations of quantifiers (which would otherwise introduce Skolem functions) *etc.* Note that this is not simply the Skolem-Lowenheim Theorem either, since the theorem is false when S mentions *every* constant, as in the set $\{\exists x.P(x)\} \cup \{\neg P(c) \mid c \in \mathcal{C}\}$. This is an example of a satisfiable set that has no standard model. As a trivial consequence of the theorem, we get

Corollary 3: *If S is finite and $\mathcal{E} \cup S \models \alpha^x_c$ for every $c \in \mathcal{C}$, then $\mathcal{E} \cup S \models \forall x.\alpha$.*

Proof: If $\mathcal{E} \cup S \not\models \forall x.\alpha$, then $\mathcal{E} \cup S \cup \{\neg\forall x.\alpha\}$ is satisfiable, and since S is finite, by the Theorem, has a standard model M. Then, by Lemma 1, for some c, $M \models \neg\alpha^x_c$, and so $\mathcal{E} \cup S \not\models \alpha^x_c$. ∎

The second theorem concerns substitutions by constants:

Theorem 4: *Let S be a set of closed wffs, let α be a wff with a single free variable x, and let H^+ be a set of constants containing those in S, those in α, and at least one constant in neither. Then for every constant $d \in \mathcal{C}$, there is a constant $c \in H^+$ such that $\mathcal{E} \cup S \models \alpha^x_d$ iff $\mathcal{E} \cup S \models \alpha^x_c$.*

Proof: It is sufficient to show that if c and d are two constants that do not appear in S or α, then

$$\mathcal{E} \cup S \models \alpha^x_d \quad \text{iff} \quad \mathcal{E} \cup S \models \alpha^x_c.$$

To prove this, first let $*$ be any bijection from $\mathcal{C}$ to $\mathcal{C}$. Let α^* mean α with c replaced by c^*. Let S^* mean $\{\alpha^* \mid \alpha \in S\}$. Finally, for any interpretation M, let M^* mean the interpretation exactly like M except that the denotation of c in M^* is changed to that of c^* in M. Then, we can prove by induction that for any α, and any assignment to variables μ,

$$M^*, \mu \models \alpha \quad \text{iff} \quad M, \mu \models \alpha^*.$$

Now to prove the theorem, suppose that $\mathcal{E} \cup S \models \alpha^x_d$, and suppose that M is any interpretation such that $M \models \mathcal{E} \cup S$. We will show that $M \models \alpha^x_c$. To do so, let $*$ be the bijection that swaps c and d and leaves all other constants unchanged. Then $(\mathcal{E} \cup S)^* = (\mathcal{E} \cup S)$, and so $M \models (\mathcal{E} \cup S)^*$. By the above, we get that $M^* \models \mathcal{E} \cup S$ and so $M^* \models \alpha^x_d$. Applying the above again, we get that $M \models (\alpha^x_d)^*$, and so $M \models \alpha^x_c$, which completes the proof. ▮

It is this theorem that will allow us to restrict our attention a finite set of constants in H^+ when we do substitutions, as we will show below. Note that the theorem is false if H^+ contains just the constants in S and α. For example, let α be $P(x)$, and S be $\{\forall z(z \neq a \supset P(z))\}$. In this case, the only constant in S or α is a, and $\mathcal{E} \cup S \not\models \alpha^x_a$, but $\mathcal{E} \cup S \models \alpha^x_b$. The theorem is also false if H^+ does not contain the constants in α. For example, let α be $R(x, b)$, and S be $\{\forall y.\forall z.(y = z) \supset R(y, z)\}$. Here, $\mathcal{E} \cup S \models \alpha^x_b$, but for every other constant c, $\mathcal{E} \cup S \not\models \alpha^x_c$.

3.2 KNOWLEDGE BASES

Since we are considering a KB containing equality, variables, and universal quantifiers, we will not be able to do simple retrieval to find out what is known about the atoms. For example, let β be the formula

$$\forall x \forall y \forall z.(x \neq y \wedge z = a) \supset R(x, z, y).$$

If *KB* contains β then we want $R(b, a, a)$ to be known. So we must first be clear about the form of KB we will be using:

Definition 4: We call a set S of formulas proper if $\mathcal{E} \cup S$ is consistent and S is a finite set of formulas of the form $\forall(e \supset \rho)$ or $\forall(e \supset \neg\rho)$.

We will be interested in KBs that are proper. Observe that as a special case, we can represent any finite consistent set of literals as a proper KB: simply replace $\rho\theta$ (or its complement) by $\forall(e \supset \rho)$ where e is of the form $\bigwedge(x_i = c_i)$. We can also represent a variety of infinite sets of literals, as the formula β does above. We are free to characterize some of the positive instances of ρ by using $\forall(e \supset \rho)$, and leave the status of the rest open. We can do the same for negative instances. We can also make a closed world assumption about a predicate if we so choose, by using both $\forall(e \supset \rho)$ and $\forall(\neg e \supset \neg\rho)$, for some e and ρ.

It might appear that proper KBs are overly restrictive, and ought to be easy to reason with. It is worth remembering that deciding whether a proper KB entails a formula is recursively unsolvable, unless the formula is restricted in some way, as we intend to do.

Although proper sets are not the same as sets of literals, they can be used to represent them in the following way:

Definition 5: Let S be any finite set of $\forall(e \supset \alpha)$ formulas as above, but not necessarily consistent. Define

$$\mathit{Lits}(S) = \{\alpha\theta \mid \forall(e \supset \alpha) \in S,\ \mathcal{E} \models e\theta\}.$$

Then we get the following:

Theorem 5: *Let S be a finite set of formulas of the above form, and let M be any standard interpretation. Then*

$$M \models S \quad \textit{iff} \quad M \models \mathit{Lits}(S)$$

Proof: For the only-if direction, observe that $\mathcal{E} \cup S$ entails every element of *Lits(S)*. Thus, because $M \models S$ and by Lemma 1, $M \models \mathcal{E}$, it follows that $M \models \mathit{Lits}(S)$.

For the if-direction, assume that $M \models \mathit{Lits}(S)$, and suppose that $\forall(e \supset \alpha) \in S$, where α is either ρ or $\neg\rho$. Assume that for some θ, $M \models e\theta$. Then by Lemma 1, $\mathcal{E} \models e\theta$, so $\alpha\theta \in \mathit{Lits}(S)$, and $M \models \alpha\theta$. Since this works for any θ, and the model is standard, we get by Lemma 1 that $M \models \forall(e \supset \alpha)$, and so M satisfies S. ▮

So S and *Lits(S)* are satisfied by the same standard interpretations (although there will be non-standard interpretations where they diverge). As corollary, we get

Corollary 6: *Let S be as above. Then $\mathcal{E} \cup S$ has a standard model iff Lits(S) is consistent.*

Proof: The only-if direction is immediate.

For the if-direction, let M be the standard interpretation whose domain is the constants $\mathcal{C}$, where each constant is interpreted as itself, where = is interpreted as identity, and where any predicate Q is interpreted as the set of tuples $\{\vec{c} \mid Q(\vec{c}) \in \mathit{Lits}(S)\}$. Since, *Lits(S)* is assumed to be consistent, we get that $M \models \mathit{Lits}(S)$, and so $M \models \mathcal{E} \cup S$ by the Theorem. ▮

3.3 ATOMIC QUERIES

Now we want to define how atomic queries will be handled for proper KBs. We will use the fact that V has already been defined for closed ewffs, and (by a simple induction argument) satisfies the following:

Lemma 7: $V[e\theta] = 1$ *iff* $\mathcal{E} \models e\theta$.

This establishes that V is logically correct for ewffs.

Definition 6: For any proper *KB*, the atomic evaluation associated with *KB* is the function V where the H (for handling quantifiers) is the set of constants mentioned in *KB*, and such that for any ground atom $\rho\theta$

$$V[\rho\theta] = \begin{cases} 1 & \text{if there is a } \forall(e \supset \rho) \in KB \\ & \quad \text{such that } V[e\theta] = 1 \\ 0 & \text{if there is a } \forall(e \supset \neg\rho) \in KB \\ & \quad \text{such that } V[e\theta] = 1 \\ \frac{1}{2} & \text{otherwise} \end{cases}$$

This function is well-defined, in that, if there were formulas $\forall(e_1 \supset \rho), \forall(e_2 \supset \neg\rho) \in KB$ such that $V[e_1\theta] = V[e_2\theta] = 1$, we would have by Lemma 7 that $\mathcal{E} \models e_1\theta \wedge e_2\theta$, and so $\mathcal{E} \cup KB \models \rho\theta \wedge \neg\rho\theta$, violating the consistency of $\mathcal{E} \cup KB$.

Furthermore, the function (as a procedure) runs in time that is no worse than linear in the size of the *KB*. Given the considerations discussed in the previous section, this settles the efficiency question as far as we are concerned: using the evaluation V associated with a KB, arbitrary closed queries can be answered efficiently.

We now turn to the correctness of V.

3.4 SOUNDNESS OF QUERY EVALUATION

We begin by showing that the evaluation associated with a *KB* always returns logically correct answers for atomic queries.

Theorem 8: *For any proper KB, the evaluation associated with KB is logically correct for ground atomic queries wrt $\mathcal{E} \cup KB$.*

Proof: For soundness, assume that $V[\rho\theta] = 1$. Then by Lemma 7, we have that $\mathcal{E} \models e\theta$, where $\forall(e \supset \rho) \in KB$, from which it follows that $\mathcal{E} \cup KB \models \rho\theta$. The case when $V[\rho\theta] = 0$ is analogous. For completeness, assume that $\mathcal{E} \cup KB \models \rho\theta$. It follows that

$$\mathcal{E} \cup KB \cup \{\forall.\wedge(x_i = c_i) \supset \neg\rho\}$$

is inconsistent, for the appropriate variables and constants. This set therefore has no standard models, and by Corollary 6, *Lits(KB)* $\cup \{\neg\rho\theta\}$ is inconsistent. Thus, $\rho\theta \in$ *Lits(KB)*, from which it follows that $V[\rho\theta] = 1$. The case when $\mathcal{E} \cup KB \models \neg\rho\theta$ is analogous. ∎

Next we show that the evaluation associated with a *KB* always returns logically sound answers for any query:

Theorem 9: *Suppose KB is proper. Then the evaluation associated with KB is logically sound for any closed formula wrt $\mathcal{E} \cup KB$.*

Proof: The proof is by induction on the length of the query. The only tricky case is when the query is $\forall x.\alpha$. If $V[\forall x.\alpha] = 0$, then for some $c \in H^+$, $V[\alpha_c^x] = 0$, and so by induction, $\mathcal{E} \cup KB \models \neg\alpha_c^x$, and so $\mathcal{E} \cup KB \models \neg\forall x.\alpha$. If $V[\forall x.\alpha] = 1$ then for every $c \in H^+$, $V[\alpha_c^x] = 1$, and so by induction, for every $c \in H^+$, $\mathcal{E} \cup KB \models \alpha_c^x$. Then, by Theorem 4, we have that for every $c \in \mathcal{C}$, $\mathcal{E} \cup KB \models \alpha_c^x$, and by Corollary 3, $\mathcal{E} \cup KB \models \forall x.\alpha$. ∎

As a simple consequence of this soundness, we also have that when the *KB* happens to have complete knowledge of some set of predicates, the evaluation is logically complete:

Corollary 10: *Suppose KB is proper, and that for every θ and every ρ taken from some set of predicates, $\mathcal{E} \cup KB \models \rho\theta$ or $\mathcal{E} \cup KB \models \neg\rho\theta$. Then the evaluation associated with KB is logically complete wrt KB for any closed formula that uses just those predicates.*

Proof: First observe that $V[\alpha] \neq \frac{1}{2}$ for any query using just those predicates (by induction on the length of the query). If $\mathcal{E} \cup KB \models \alpha$, then since *KB* is proper, $\mathcal{E} \cup KB \not\models \neg\alpha$, and so by soundness, $V[\alpha] \neq 0$ and so $V[\alpha] = 1$. The case where $\mathcal{E} \cup KB \models \neg\alpha$ is analogous. ∎

This theorem shows that for complete knowledge represented as above, evaluation provides an efficient and logically correct deductive reasoning procedure.

However, as we already argued, we cannot expect to have logical correctness when knowledge is incomplete. In the next section, we show that we do get it for the special case of queries in normal form.

3.5 NORMAL FORM

This is the normal form we will be using:

Definition 7: A set S of closed formulas is logically separable iff for every consistent set of ground literals L, if $L \cup \{\alpha\}$ is consistent for every $\alpha \in S$, then $L \cup S$ has a standard model.

Definition 8: The normal form formulas $\mathcal{NF}$ is the least set such that

1. if α is a ground atom or ewff, then $\alpha \in \mathcal{NF}$;
2. if $\alpha \in \mathcal{NF}$, then $\neg\alpha \in \mathcal{NF}$;
3. if $S \subseteq \mathcal{NF}$, S is logically separable, and S is finite, then $\wedge S \in \mathcal{NF}$;
4. if $S \subseteq \mathcal{NF}$, S is logically separable, and for some α, $S = \{\alpha_c^x \mid c \in \mathcal{C}\}$, then $\forall x.\alpha \in \mathcal{NF}$.

Before explaining how the definition works, we prove the main theorem:

Theorem 11: *Suppose KB is proper. Then the evaluation associated with KB is logically complete for any normal form formula wrt $\mathcal{E} \cup KB$.*

Proof: The proof is by induction on the length of the query. For atoms, ewffs, and negations, the argument is clear.

For conjunctions (other than ewffs), if $\mathcal{E} \cup KB \models (\alpha \wedge \beta)$ then $\mathcal{E} \cup KB \models \alpha$ and $\mathcal{E} \cup KB \models \beta$, and so by induction $V[\alpha] = 1$ and $V[\beta] = 1$, meaning $V[\alpha \wedge \beta] = 1$. If on the other hand, $\mathcal{E} \cup KB \models \neg(\alpha \wedge \beta)$ then $\mathcal{E} \cup KB \cup \{\alpha, \beta\}$ has no standard models. By Theorem 5, *Lits(KB)* $\cup \{\alpha, \beta\}$ has no standard models either. Since $(\alpha \wedge \beta) \in \mathcal{NF}$, either *Lits(KB)* $\cup \{\alpha\}$ or *Lits(KB)* $\cup \{\beta\}$ is inconsistent, and has no standard models. By Theorem 5, either $\mathcal{E} \cup KB \cup \{\alpha\}$ or $\mathcal{E} \cup KB \cup \{\beta\}$ has no standard models. By Theorem 2, $\mathcal{E} \cup KB \models \neg\alpha$ or $\mathcal{E} \cup KB \models \neg\beta$, and so by induction, $V[\alpha] = 0$ or $V[\beta] = 0$, implying $V[\alpha \wedge \beta] = 0$.

For quantifications, if $\mathcal{E} \cup KB \models \forall x.\alpha$ then for every c, $\mathcal{E} \cup KB \models \alpha^x_c$, and so by induction, $V[\alpha^x_c] = 1$, and then $V[\forall x.\alpha] = 1$. If on the other hand, $\mathcal{E} \cup KB \models \neg\forall x.\alpha$ then $\mathcal{E} \cup KB \cup \{\forall x.\alpha\}$ has no standard models. By Theorem 5, *Lits(KB)* $\cup \{\forall x.\alpha\}$ has no standard models either. Since $\forall x.\alpha \in \mathcal{NF}$, for some $c \in \mathcal{C}$, *Lits(KB)* $\cup \{\alpha^x_c\}$ is inconsistent, and has no standard models. By Theorem 5, for some $c \in \mathcal{C}$, $\mathcal{E} \cup KB \cup \{\alpha^x_c\}$ has no standard models. By Theorem 2, for some $c \in \mathcal{C}$, $\mathcal{E} \cup KB \models \neg\alpha^x_c$, and therefore by Theorem 4, for some $c \in H^+$, $\mathcal{E} \cup KB \models \neg\alpha^x_c$. Then, by induction, we have that for some $c \in H^+$, $V[\alpha^x_c] = 0$ and so $V[\forall x.\alpha] = 0$. ∎

This theorem shows that as long as the query is in normal form, we have an efficient deductive reasoning procedure for first-order KBs with incomplete knowledge that is guaranteed to be logically correct. In other words, we can evaluate a query to determine if it or its negation is entailed, and always get answers that are logically correct.

3.5.1 Examples of normal form

So see how the definitions of logically separable and normal form work, it is best to start with the propositional case (and ignore "standard" in the definition). Notice that any literal will be in normal form. If L and S are both consistent sets of literals and $L \cup S$ is inconsistent, then they contain complementary literals, and so $L \cup \{\rho\}$ is inconsistent for some $\rho \in S$. It follows that S is logically separable, according to the definition. So if S is a finite consistent set of literals (including the empty set), $\wedge S \in \mathcal{NF}$ and $\neg\wedge S \in \mathcal{NF}$. Thus $\mathcal{NF}$ contains all non-tautologous clauses and their complements. On the other hand, $\{p, \neg p\}$ is not logically separable since $\{q, p, \neg p\}$ is inconsistent, but $\{q, p\}$ and $\{q, \neg p\}$ are both consistent. So $\mathcal{NF}$ does not contain tautologous clauses and their complements.

Now the next question is which conjunctions of non-tautologous clauses will be in normal form. Consider the set $S = \{(p \vee \neg q), (\neg p \vee r)\}$. Let L be $\{q, \neg r\}$. Then $L \cup S$ is inconsistent, but both $L \cup \{(p \vee \neg q)\}$ and $L \cup \{(\neg p \vee r)\}$ are consistent, so S is not logically separable. But consider $S' = S \cup \{(\neg q \vee r)\}$. In this case, it can be shown that any consistent set of literals that is inconsistent with S' will also be inconsistent with one of the clauses in S'. The result is that $\wedge S' \in \mathcal{NF}$ and $\neg\wedge S' \in \mathcal{NF}$. As we will see more generally below, to guarantee separability in the propositional case, it is sufficient that a set of non-tautologous clauses be closed under Resolution.

To see what work this constraint will do, observe that for any KB and any α and β, we have that

$$\text{if } KB \models (\alpha \wedge \beta), \text{ then } KB \models \alpha \text{ and } KB \models \beta.$$

This suggests that if V gets the correct value for α and β, it will get the correct value for $(\alpha \wedge \beta)$. But the following does *not* hold in general:

$$\text{if } KB \models \neg(\alpha \wedge \beta), \text{ then } KB \models \neg\alpha \text{ or } KB \models \neg\beta,$$

for example, as above when $KB = \{q, \neg r\}$, $\alpha = (p \vee \neg q)$, and $\beta = (\neg p \vee r)$. However, the above property *does* hold when the query is in $\mathcal{NF}$, and this will give us completeness for all normal form queries.

In the first order case, the considerations are similar, with universal quantification behaving somewhat like infinite conjunction. For any KB and any α we have that

$$\text{if } KB \models \forall x.\alpha, \text{ then } KB \models \alpha^x_c, \text{ for every } c.$$

But the following does not hold in general:

$$\text{if } KB \models \neg\forall x.\alpha, \text{ then } KB \models \neg\alpha^x_c \text{ for some } c.$$

A simple example is when α is the formula

$$(\neg P(x) \wedge \exists y.P(y)),$$

or expressed in terms of existentials, when the query is $\exists x(P(x) \vee \forall y.\neg P(y))$. This existential is entailed by the empty KB, but no substitution instances are (nor, for that matter, are disjunctions of substitution instances). The implication is that even if V gets the correct value for every α^x_c, it need not get the correct value for $\forall x.\alpha$. The definition of $\mathcal{NF}$ rules out such formulas as queries.

One subtlety in the definition is the requirement for the model of $L \cup S$ to be *standard*. When S is finite, as in the case of conjunction, this imposes no additional constraint. But its importance can be seen from the example of α above. Here $S = \{\alpha^x_c \mid c \in \mathcal{C}\}$, and when $L \cup \{\alpha^x_c\}$ is consistent for every c, $L \cup S$ does unfortunately have a model: it is a non-standard one where $P(c)$ comes out false for every c, but $\exists x.P(x)$ comes out true. However, S is not logically separable, since $L \cup S$ cannot have a *standard* model, as S itself does not have one. Consequently, neither $\forall x.\alpha$ nor its negation are in normal form, as desired.

3.5.2 The expressive range of $\mathcal{NF}$

As a final topic, we consider the expressive range of $\mathcal{NF}$. We can prove that in the propositional sublanguage, the restriction to normal form is without loss of expressive power:

Lemma 12: *Suppose S is a finite set of ground clauses that is closed under Resolution, in the sense that*

> *If $\alpha \in S$, $\beta \in S$, and α and β resolve to γ, then either γ is tautologous or there is a $\delta \in S$ such that $\delta \subseteq \gamma$.*

Then S is logically separable.

Proof: Suppose that $L \cup S$ is inconsistent for some consistent set of literals L. We will show that for some $\alpha \in S$, $L \cup \{\alpha\}$ is inconsistent. Let $\{d_1, d_2, \ldots, d_n\}$ be a Resolution refutation of $L \cup S$, with no tautologies, and with all uses of L moved to the end. Because L is consistent, there must be at least one clause of S used. So there will be some k, $1 \leq k \leq n$, where d_k is derivable using only the clauses in S, and d_k uses just the literals in L to get to the empty clause. Thus, $L \cup \{d_k\}$ is inconsistent. Since S is closed under Resolution, there is an $\alpha \in S$, such that $\alpha \subseteq d_k$, and so $L \cup \{\alpha\}$ is inconsistent too. ∎

Theorem 13: *In the propositional sublanguage, for every $\alpha \in \mathcal{L}$, there is $\alpha' \in \mathcal{NF}$ such that $\models (\alpha \equiv \alpha')$.*

Proof: Consider the following operation on α: convert α to CNF, and starting with this set of clauses, run Resolution repeatedly on the resulting clauses, deleting any tautologous or subsumed ones until no new clauses are generated. Call the (finitely many) resulting clauses A. Since each element of A is non-tautologous, as noted above, $A \subseteq \mathcal{NF}$. Further, A is closed under Resolution, and so by the Lemma, A is logically separable. Now let α' be $\wedge A$. Then, $\alpha' \in \mathcal{NF}$, and $\models (\alpha \equiv \alpha')$, which completes the proof. ∎

The formula $\alpha' \in \mathcal{NF}$ used in the proof of this theorem is in what is called *Blake Canonical Form* (BCF) [2]. Using later terminology (due to Quine), it is the conjunction of the non-tautologous prime implicates of α. Note, however, that while $\mathcal{NF}$ includes BCF, it goes beyond it, in that it is closed under negation, and has formulas of arbitrary alternations of $\wedge$ and $\vee$. As a very simple example, suppose that α and β are in BCF and share no atoms. Then it is easy to show that $\{\neg\alpha, \neg\beta\}$ is logically separable, and so $(\alpha \vee \beta) \in \mathcal{NF}$.

I have as yet been unable to prove or disprove that the above theorem generalizes to the first-order case. To see some of the complications, consider, for example, the formula $\exists x.\neg R(a, x) \wedge R(x, b)$ as a query. This is in prenex form, has a matrix that is both in CNF and in DNF, and does not appear to involve hidden valid formulas or their negations. This seemingly innocuous formula is not in $\mathcal{NF}$, however. To see why it presents difficulties, suppose we have a *KB* where *Lits(KB)* is $\{\neg R(a, a), R(b, b)\}$. Although this KB does not entail any instance of the query, by reasoning by cases with $R(a, b)$, we can see that it entails the existential.[6] Clearly we would not be able to perform the deduction in this case by finding an appropriate substitution, and so our evaluation-based reasoning method fails on this formula.

It is not, however, a counterexample to the first-order version of the above theorem because there is a formula equivalent to it that is in $\mathcal{NF}$. It is easiest to start with the dual, $\forall x.R(a, x) \vee \neg R(x, b)$. Observe that this formula is logically equivalent to

$$\forall x.(R(a, x) \vee \neg R(x, b)) \wedge (R(a, a) \vee \neg R(b, b))$$

which, with some effort, can be shown to be in $\mathcal{NF}$. So the negation of this formula is an existential that is equivalent to the query and is in $\mathcal{NF}$.[7]

Although I cannot prove that every first-order formula has an equivalent normal form variant, it is useful to consider some special cases guaranteed to be in normal form. For example, we have

Theorem 14: *If S is proper, then $\wedge S \in \mathcal{NF}$.*

Proof: It is not hard to see that $S \subseteq \mathcal{NF}$. To see that S is logically separable, suppose that for some consistent set of literals L, $L \cup S$ has no standard model. Then by Theorem 5, $L \cup$ *Lits(S)* has no standard model. Thus, there is $\alpha\theta \in$ *Lits(S)* whose complement is L. This implies that $L \cup \{\forall(e \supset \alpha)\}$ is inconsistent, for the appropriate e. ∎

Another special case is as follows:

Definition 9: Two literals are <u>*conflict-free*</u> iff either they have the same polarity, or they use different predicates, or they use different constants at some argument position.

Theorem 15: *If all the literals in α are conflict-free, then $\alpha \in \mathcal{NF}$.*[8]

Proof: (sketch) Put α into a prenex CNF form. It suffices to show that if we have clauses $\gamma_1, \ldots, \gamma_n$, all of whose literals are conflict-free, then the set of instances of $\{\gamma_1, \ldots, \gamma_n\}$ is logically separable. ∎

[6] This is variant of the "3 block problem" discussed in [15]. In our case, from the given KB, we know that the x in question must be a or b, but we cannot say which.

[7] Observe that the conversion to $\mathcal{NF}$ required conjoining a non-subsumed entailed clause to the universal formula, as we do with the BCF. To prove the theorem, it would suffice to show that only finitely many such clauses, perhaps with new variables and equalities, are ever required.

[8] A literal appears in α if the corresponding atom appears within the scope of an appropriate number (odd or even) of negation operators.

As a further special case of this theorem, if we have a query where every predicate letter appears only positively or only negatively, we are guaranteed to be in normal form, and so to get logically correct answers.

4 CONCLUSION

In this paper, we have *not* proposed a new way of doing deductive reasoning. Rather, we have proved some properties of an existing and well known method (in the propositional case, at least) for efficiently dealing with incomplete knowledge: to determine whether or not a formula is known, we evaluate it, reducing the question to what is known about (instances of) its atomic components. Of course, other more general methods do exist for handling incomplete knowledge efficiently, and our results say nothing about the sort of incomplete knowledge that arises from disjunctions or existentials in a KB. For these, one needs to use a more complex deductive procedure to guarantee efficiency, such as the one described in [9, 16]. I suspect that similar completeness theorems could be proved about that system as well.

The results here can be thought of as one possible approach to the expressiveness / tractability tradeoff [12], especially for large KBs: we allow a very limited (but arguably, quite useful) form of incompleteness in the KB, but to preserve tractability, we insist that the query be in a certain normal form.

In my opinion, it is *not* useful to think of the results here as suggesting some sort of two-phased approach to reasoning with incomplete knowledge: first, perform query optimization by putting the query into $\mathcal{NF}$, and second, perform evaluation-based reasoning. The first phase is too difficult to do in an online automated way. A good analogy is with programs that terminate properly. We would not expect a system that *executes* programs to first filter out those programs that run forever. Rather, we make it *our* responsibility to write programs that terminate properly, and where necessary prove (by hand, usually) that they terminate before even submitting them for execution.

As to future work, I believe that it would not be overly difficult to generalize our KB to include predicates that are *defined* in terms of more basic ones (in a stratified way), preserving both correctness and efficiency. Incorporating Skolem constants (null values) in the KB is another possible extension. It would also be useful to characterize other easy-to-check special cases of normal form queries. But the more pressing open question, perhaps, is the expressive range of these queries: does every closed formula have an equivalent normal form? Should the normal form defined here be shown to be less than fully general, the obvious next step would be to look for a more expressive one that remains tractable.

Acknowledgments

The research reported here was done while the author was visiting at Simon Fraser University and York University. I most gratefully acknowledge my Engine Room cronies Jim Delgrande and Pat Dymond for technical discussions, computer support, and fun and games. I would also like to thank David Mitchell, Gerhard Lakemeyer, and Ray Reiter for helpful comments, and John Funge for interesting conversations about real-valued fluents and intervals, which originally inspired the work here. Finally, I acknowledge the Natural Sciences and Engineering Research Council of Canada for the moolah.

References

[1] Belnap, N., "A useful four-valued logic", in *Modern uses of multiple-valued logic*, J. Dunn and G. Epstein, eds., Reidel Publishing Co., 1977, pp. 8–37.

[2] Blake, A., *Canonical expressions in Boolean algebra*, Ph. D. thesis, University of Chicago, 1938.

[3] Cadoli. M. and Schaerf, M., "Approximate reasoning and non-omniscient agents", in *Proc. TARK 92*, Los Altos, CA, 1992, Morgan Kaufmann Publishers, Inc., pp. 169–183.

[4] Dunn, M., "Intuitive semantics for first-degree entailments and 'coupled trees'", *Philosophical Studies*, **29**, pp. 149–168, 1976.

[5] Ginsberg. M. "Multivalued logics: a uniform approach to reasoning in Artificial Intelligence", *Computational Intelligence*, **4**, pp. 265–316, 1988.

[6] Kleene, S. "On a notation for ordinal numbers", *Journal of symbolic logic*, **3**, pp. 150-155, 1938.

[7] Hogg, T., Huberman, B., Williams, C. (eds.), *Frontiers in problem solving: phase transitions and complexity*, special issue of *Artificial Intelligence*, **81**, March 1996.

[8] Kautz, H., McAllester, D., Selman, B. "Encoding plans in propositional logic", in *Proc. of KR-96*, Cambridge, MA, 1996.

[9] 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.

[10] Levesque, H., "A logic of implicit and explicit belief", in *Proc. AAAI-84*, pp. 198–202, Austin, TX, 1984.

[11] Levesque, H. "Making believers out of computers," *Artificial Intelligence*, **30**, 1986, pp. 81–108.

[12] Levesque, H., Brachman, R. "Expressiveness and tractability in knowledge representation and reasoning," *Computational Intelligence*, **3**, 2, May 1987, pp. 78–93.

[13] Levesque, H., Lakemeyer, G. *The Logic of Knowledge Bases: Foundations of a Functional Approach to Knowledge Representation*, technical monograph, coming soon.

[14] McCarthy, J, "Programs with common sense," in *Semantic Information Processing*. M. Minsky, *ed.* Cambridge, MA: The MIT Press, 1968, pp. 403–418. Reprinted in *Readings in Knowledge Representation.* R. J. Brachman and H. J. Levesque, *eds.* San Mateo, CA: Morgan Kaufmann Publishers, Inc., 1985, pp. 299–307.

[15] Moore, R., "The role of logic in knowledge representation and commonsense reasoning," in *Proceedings of AAAI-82*, Pittsburgh, PA, pp. 428–433, 1982.

[16] Patel-Schneider, P., "A decidable first-order logic for knowledge representation," *Proceedings of IJCAI-85*, Los Angeles, CA, pp. 455–458, 1985.

[17] Selman, B., Levesque, H., and Mitchell, D. "A new method for solving hard instances of satisfiability," in *Proc. of AAAI-92*, San Jose, Aug. 1992, pp. 440–446.

[18] Urquhart, A., "Many-valued logic", in *Handbook of philosophical logic, vol. III*, D. Gabbay and F. Guenthner (eds.), Reidel Publishing Company, 1986.

[19] Van Fraasen, B., "Singular terms, truth-value gaps, and free logic", *Journal of philosophical logic*, **63**, pp. 481–495, 1966.

[20] Vassiliou, Y. *A formal treatment of incomplete information in database management*, Ph. D. thesis, Dept. of Computer Science, University of Toronto, 1980.

A Strategy for Revising Default Theory Extensions

Mary-Anne Williams
Information Systems
School of Management
The University of Newcastle
NSW 2308, Australia
maryanne@infosystems.newcastle.edu.au

Grigoris Antoniou
School of Computing and Information Technology
Griffith University
QLD 4111, Australia
ga@cit.gu.edu.au

Abstract

Default reasoning is used to enrich a body of information with plausible conjectures. The ability to draw such conjectures facilitates decision making in the absence of complete information so rather than remaining indecisive a reasoning agent can fill in missing information on the basis of its default expectations. *Exten* is an object-oriented default logic system implemented in C^{++}. It calculates not only all the Reiter extensions, but also the justified, constrained and modified extensions of a given default theory via the construction of a process tree. Its performance is enhanced via the use of stratification.

One of the problems with default logics in general, and *Exten* in particular, is that if the original default theory changes due to the acquisition of new information then it must not only recalculate all the extensions from scratch, but it is incapable of adding new information that is inconsistent with the set of facts because it does not possess belief revision capabilities.

In this paper we provide a computational strategy for revising a default theory. This strategy creates the same set of extensions that a complete recalculation of extensions from the updated set of facts would result in. *Exten* is currently being augmented with the proposed strategy. The essential idea of the strategy is that it guides the minimal pruning of the process tree, from which the new extensions are subsequently constructed. This method can be thought of as a form of process tree repair. Nothing is gained in the worst case because the whole process tree might necessarily be lopped off, i.e. pruned back to the root, and then a whole new process tree constructed from scratch. But since default logic is computationally expensive, it is abundantly clear, that there exist numerous cases where there is a substantial computational saving to be made by pruning. Furthermore, almost nothing is lost by trying to prune from a computational point of view because it can be immediately determined if the whole process tree should be lopped. It is noteworthy that the stratification of default rules can not only improve the generation of a process tree from scratch, but it can also improve the performance of the revision strategy!

1 INTRODUCTION

Nonmonotonic Reasoning was introduced to overcome several difficulties that arise when an agent's knowledge is incomplete. Techniques in nonmonotonic reasoning provide mechanisms that allow the agent to make useful defeasible conjectures which would not be derivable using classical logic. For instance, nonmonotonic conjectures are often based on the absence of information, e.g. if want to locate my copy of *Ulysses* and I am unaware that a friend has taken the liberty of borrowing it without asking, then it would be reasonable to conjecture that it is still on my bookshelf where I left it. Such default conjectures are defeasible, and more defeasible than the information I strongly believe to be the case, such as James Joyce is the author of *Ulysses*. Nonmonotonic reasoning is used in a wide variety of applications; from the common database to the more sophisticated intelligent information system.

The conjectures are made in the face of incompleteness on the understanding that if new information is acquired that is in conflict with them then they should no longer be taken into consideration. Despite this understanding surprisingly little work has been done on building computational models for revising nonmonotonic theories.

For default logic Williams and Foo [25] developed a theoretical framework for changing a default theory, but almost no work has been conducted on developing computational strategies for implementing the revision of default logic theories.

In this paper we develop a strategy for revising a default theory. The proposed strategy can be readily modified to revise several variants of default logic, for example, constrained, modified and defaults with priority. The proposed algorithms are currently being used to augment *Exten* a default reasoning system [9] [3] which was developed under the umbrella of the CIN project; a project seeking to develop an intelligent information management toolkit [4]. The project's aim is to build an integrated, domain-independent tool for default reasoning and belief revision which will be fully accessible on the WWW[26]. We intend to apply the system in various applications which require reasoning with incomplete and changing information. Such forms of reasoning are typically required in decision making based on semi-structured or unstructured information. Also, representations using default rules are well suited as a compact representation language for program specifications, and as a language for requirements engineering (since the representation of conflicting information is supported) [16].

2 DEFAULT REASONING

Default logic [19] provides a formal theory of reasoning based on default rules, and it is is one of the most prominent approaches of nonmonotonic reasoning. One of the main problems with its applicability is that it is computationally harder than classical logic [17, 12]. One of the most promising ways of easing this problem and developing powerful implementations is to split a default theory into smaller parts and compute extensions in a modular, "local" way. One approach is to use stratification of default theories [5, 6, 7] which splits a default theory in a way that preserves extensions. Stratification is based on the following idea: If the consequent of a default δ may be used by another default δ', then seek to apply δ before δ'. This way we obtain information about potential orderings of defaults. Using this idea, a powerful system for default reasoning was developed at the University of Kentucky [8]. A similar idea was presented in [21].

The strategy for revising a default logic theory presented in this paper provides a sound theoretical foundation for revising the extensions generated by the default reasoning system *Exten.*

2.1 DEFAULT LOGIC

We assume that the reader is familiar with the notation and basic notions of classical logic. In this paper we restrict attention to propositional default theories, as does Cholewinski; a generalization to closed defaults in first-order logic is straightforward. A propositional *signature* is determined by a set of propositional atoms U. For a formula φ, the set of propositional atoms occurring in φ is denoted by $Prop(\varphi)$. This notion extends to sets of formulae in a natural way.

A *default* δ has the form $\frac{\varphi:\psi_1,\ldots,\psi_n}{\chi}$ with propositional formulae φ, $\psi_1,\ldots,\psi_n$, χ $(n \geq 1)$. φ is the *prerequisite* $pre(\delta)$, $\psi_1,\ldots,\psi_n$ the *justifications* $just(\delta)$, and χ the *consequent* $cons(\delta)$ of δ. A *default theory* T is a pair (W, D) consisting of a set of formulae W (the set of *facts*) and a countable set D of defaults. In this paper we consider finite sets D of defaults (which is acceptable from the practical viewpoint, of course).

Let $\delta = \frac{\varphi:\psi_1,\ldots,\psi_n}{\chi}$ be a default, and E a deductively closed set of formulae. We say that δ *is applicable to* E iff $\varphi \in E$, and $\neg\psi_1,\ldots,\neg\psi_n \notin E$.

Let $T = (W, D)$ be a default theory and $\Pi = (\delta_0,\ldots,\delta_m)$ a finite sequence of defaults from D without multiple occurrences (modelling an application order of defaults from D). We denote by $\Pi(k)$ the initial segment of Π of length k $(k \leq m+1)$, and by $Def(\Pi)$ the set of defaults occurring in Π.

- $In(\Pi) = Th(W \cup \{cons(\delta) \mid \delta \text{ occurs in } \Pi\})$[1].
- $Out(\Pi) = \{\neg\psi \mid \psi \in just(\delta), \delta \text{ occurs in } \Pi\}$.
- Π is called a *process of* T iff δ_k is applicable to $In(\Pi(k))$, for every $k \leq m$.
- Π is *successful* iff $In(\Pi) \cap Out(\Pi) = \emptyset$, otherwise it is *failed*. Note that in case Π is successful, every justification ψ of a default in Π is consistent with $In(\Pi)$.
- Π is *closed* iff every default that is applicable to $In(\Pi)$ already occurs in Π.

[1]where $Th(M)$ denotes the deductive closure of the set M of formulae.

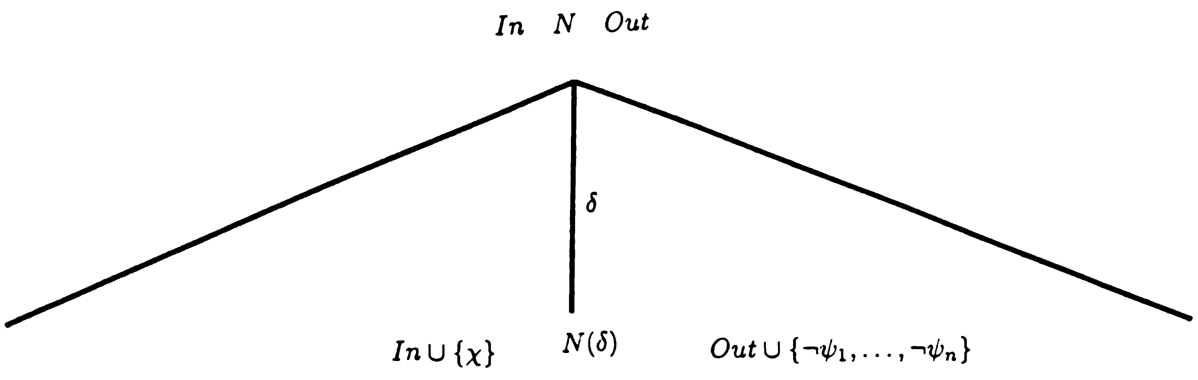


Figure 1: **The Expansion of a Node in the Process Tree by** *Exten*

It was shown in [2] that Reiter's original definition of extensions is equivalent to:

> A set of formulae E is an *extension* of a default theory T iff there is a closed and successful process Π of T such that $E = In(\Pi)$.

2.2 STRATIFICATION

Let D be a finite set of defaults. A function ρ assigning a natural number to every default from D is a *stratification function*, iff for any defaults $\delta, \delta' \in D$ the following condition holds:

If $Prop(cons(\delta)) \cap Prop(\delta') \neq \emptyset$
then $\rho(\delta) \leq \rho(\delta')$[2].

An alternative, equivalent description of this property.

If $Prop(cons(\delta)) \cap Prop(pre(\delta')) \neq \emptyset$
then $\rho(\delta) \leq \rho(\delta')$.

If $Prop(cons(\delta)) \cap Prop(just(\delta')) \neq \emptyset$
then $\rho(\delta) \leq \rho(\delta')$.

If $Prop(cons(\delta)) \cap Prop(cons(\delta')) \neq \emptyset$
then $\rho(\delta) = \rho(\delta')$.

By assigning a number to each default, ρ decomposes D into layers (strata) $D_1, \ldots, D_k$ of ascending value under ρ. We denote by $Stratum_{\rho=k}(D)$ the stratum D_i such that $\rho(\delta) = k$ for all $\delta \in D_i$. A default theory $T = (W, D)$ is called *stratified* iff W is consistent, $Prop(W) \cap Prop(cons(D) \cup just(D)) = \emptyset$, and there exists a stratification function for D.

[2]where $Prop(\delta)$ denotes the set of all propositional atoms occurring in δ.

Essentially, a stratified function is based on the idea that should the information provided by the default δ be used by the default δ', then we should seek to apply δ before δ'. The following result from [5] shows that decomposition of a set of defaults according to a stratification function preserves the extensions.

For example, consider the following defaults:

$$\delta_1 = \frac{true : p}{p}, \delta_2 = \frac{p : q}{q}, \delta_3 = \frac{r : \neg p}{s}, \delta_4 = \frac{true : true}{\neg p}.$$

According to the definition of a stratification function, δ_2 and δ_3 make use of information in the consequent of δ_1 and δ_4 (because of the propositional atom p), but not vice versa. So, $\rho(\delta_2)$ and $\rho(\delta_3)$ may be strictly greater than $\rho(\delta_1)$ and $\rho(\delta_4)$, but certainly not less. On the other hand, by definition, $\rho(\delta_1)$ must equal $\rho(\delta_4)$. Thus one possible stratification function is:

$$\rho(\delta_1) = \rho(\delta_4) = 1, \rho(\delta_2) = \rho(\delta_3) = 2.$$

This function creates two strata. Another stratification function is

$$\rho'(\delta_1) = \rho'(\delta_4) = 1, \rho'(\delta_2) = 2, \rho'(\delta_3) = 3$$

which creates three strata. The algorithms used in implementations seek to decompose a default theory into as fine a strata as possible. In our example, ρ' is preferable to ρ. The discussion of these algorithms is beyond the scope of this paper.

Regarding implementation, Cholewinski proposed a more sophisticated way of computing extensions than the linear order provided by a stratification function [6]. Instead, he used the precedence information provided by the definition of stratification to define a par-

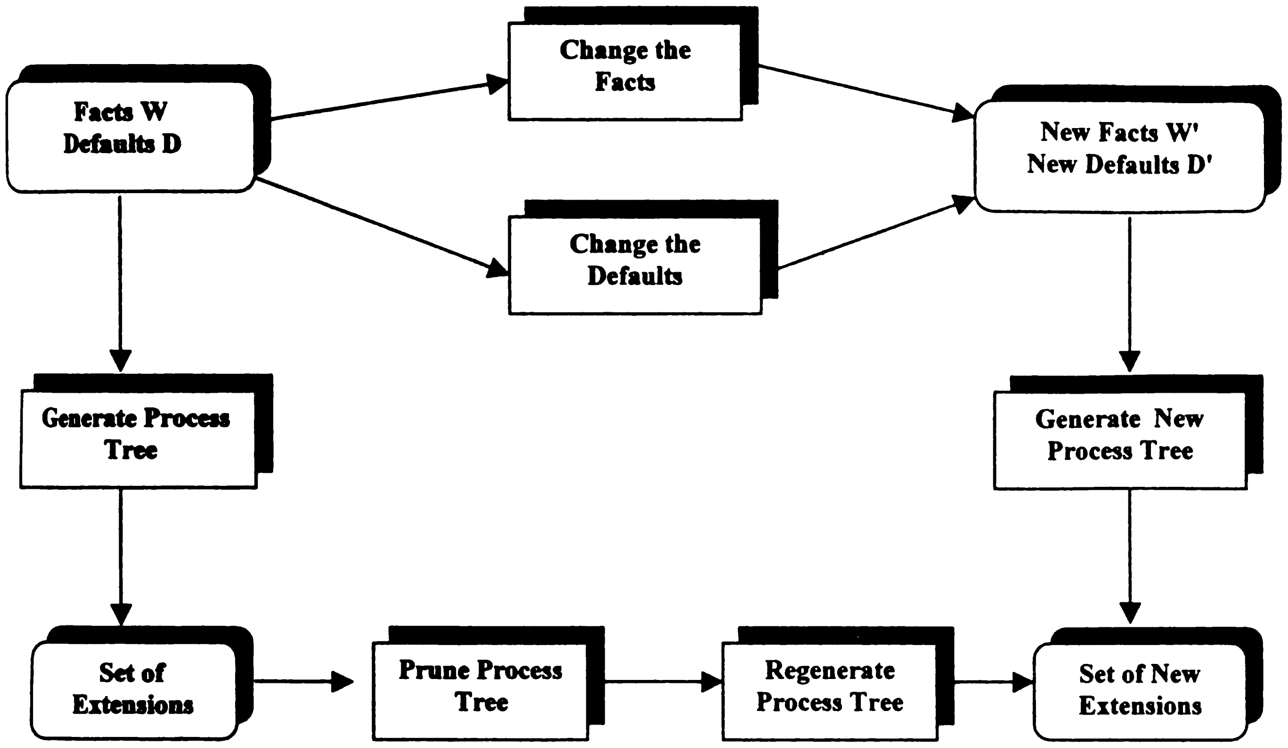


Figure 2: **Changing a Default Theory**

tially ordered set. This way, the extensions of some strata can be computed in parallel, and this can lead to a drastic improvement in performance. For example, if we consider Reiter's default logic and take T to consist of the defaults $\delta_1 = \frac{true:p}{p}, \delta_2 = \frac{p:q}{q}, \delta_3 = \frac{r:\neg p}{s}, \delta_4 = \frac{true:true}{\neg p}$, then δ_1 and δ_4 must be applied and in the same stratum, but there are no further constraints on the application of δ_2 and δ_3. Therefore they should be treated independently from one another, essentially in parallel.

Stratification is the main underlying principle in our implementation of a default reasoning system. The prototype, *Exten*, implemented in C^{++} is described in the next subsection.

2.3 *EXTEN*

Exten uses an intelligent forward chaining presentation of the default logics that reduces the search space to that of backward chaining [3, 9], i.e from the naive $n!$ to 2^n. The core algorithm is given below:

```
PROCEDURE Compute-Ext(Π, Drest, Dout)
BEGIN
   NotClosed := false;
   M := {δ ∈Drest | pre(δ)∈In(Π)} = {δ_1,...,δ_n};
   IF M = ∅ THEN
      IF IsClosed(Π, Dout) THEN
         ext := ext ∪ {In(Π)}
      FI
   ELSE
      FOR i := 1 TO n DO
         Drest := Drest - {δ_i};
         IF In(Π)⊬ Out(Π) THEN
            Compute-Ext(Π∘δ_i, Drest, Dout);
            NotClosed := true;
         FI;
         Dout := Dout ∪ {δ_i}
      END;
      IF NotClosed = false AND
         IsClosed(Π, Dout)
      THEN ext := ext ∪ {In(Π)}
      FI
   FI
END
```

The initial call of *Compute-Ext* uses the empty process for Π, the set of all available defaults D for *Drest*, and the empty set for *Dout*. The variable *ext* collects all extensions of T.

The following remarks are in order. First, if a default δ_i is applicable at the current node of the process tree, then it is deleted from the set of available defaults *Drest*.

Once a default rule, δ_i is blocked it will remain blocked even if further defaults are applied since $In(\Pi)$ grows monotonically with Π, therefore we do not lose any extension by removing it from the set of available defaults. The only catch is that when we are testing for the closure of a process, we must test the applicability of the 'dropped' defaults from *Dout*.

Consider the example $T = (W, D)$, where $W = \emptyset$ and $D = \{\delta_0 = \frac{true:a}{a}, \delta_1 = \frac{true:b}{b}, \delta_2 = \frac{true:c}{c}, \delta_3 = \frac{true:d}{d}\}$. The process tree is found in Figure 1. To illustrate how the procedure works, let us discuss part of the generation of this tree.

First the leftmost path is generated, leading to an extension. Then we look for an alternative, so we have to backtrack by two stages; then we can apply δ_3 instead of δ_2. Note that, by definition of *Compute-Ext*, $\delta_2 \notin$ *Drest* (which currently is empty), yet we have to check it for not being blocked (which is true in our case). If we had omitted this test then we would have concluded wrongly that $Th(\{a, b, d\})$ is an extension.

3 CHANGING DEFAULT THEORIES

Changing a default theory in our context amounts to pruning the process tree. Previously blocked defaults may become applicable as a result of the pruning. Consequently the pruned tree must subsequently be subjected default rule closure.

Contracting facts may lead to the inapplicability of a previously applied default rule because its prerequisite may have been removed in the contraction. Adding facts may block a default rule rendering it inapplicable because newly introduced facts may be in conflict with default rule justifications and consequents, which in turn may lead to the prerequisites of previously applicable defaults being blocked. Adding a fact that is inconsistent with the current set of facts will involve both the contraction and addition of facts, and may lead to the inapplicability of defaults due to problems with prerequisites, justifications and/or consequents.

In keeping with the principle of Minimal Change a *minimal* pruning is desirable. It is easily shown that a unique minimal pruning exists, and that our proposed strategy produces the unique minimal pruning. In particular, our algorithms minimally prune a given process tree. i.e. the process tree is not pruned beyond the nodes that represent the first default that was previously applied that is not sanctioned by the change to the default theory at its current position in the process tree (it may of course become applicable after the application of other defaults).

Although motivated by the AGM paradigm, the revision strategy decribed herein lies somewhat outside it mainly because AGM operators are not designed for default theories but classical theories. Furthermore from a more practical point of view an epistemic entrenchment [11] (or a system of spheres[13]) may not be derived from the order in which the default rules were applied during the construction of the process tree in a natural way.

3.1 CHANGING THE FACTS

The proposed strategy performs the minimal pruning of a process tree. The pruned process tree can then be used to generate the set of new extensions. We presume the existence of a belief revision algorithm that can revise the facts W, such as an adjustment [23], maxi-adjustment [24], or conditionalization [20]. In this section we provide an algorithm for determining the set of new extensions without the need to recalculate them from scratch in general.

Essentially nodes are checked in the process tree for continued validation under the pressure of change. In other words, the strategy determines the largest subtree of the process tree that could exist under the new root (the changed facts W), i.e. the subtree from the old root that would be a subtree of the new root's process tree if the extensions were calculated from scratch.

Since $\Pi(k)$ for all k are already computed the subtree validation process can be performed top-down, bottom-up or by using a more efficient strategy akin to a binary search. These strategies are also available for AGM revision [24] and possibilistic logic [?] as well.

There are four cases of change to consider:

- contraction of information,
- denial of information,
- incorporation of consistent information, and
- incorporation of inconsistent information.

Although contraction and denial are related, denial is the stronger. Contracting a fact, α, from the set of facts, W, alone may not be enough to stop α from it reappearing as a default fact after the applications of defualts in D. Denial ensures that such a fact is not in any extension.

3.1.1 Contracting Information

By contraction we mean the removal of a sentence α from the set of facts of the default theory. This does not necessarily mean that α will not included in extensions of the updated default theory, since it might be reintroduced by the application of defaults. In fact α may end up being included in all extensions of the updated theory.

Formally, given a default theory $T = (W, D)$ we define $T - \alpha = (W_\alpha^-, D)$. In the following we discuss how the extensions of the updated default theory may be computed while maintaining part of the calculations made for the extensions of T.

The contraction of information from W may result in prerequisites of generating defaults of a particular extension being lost, and so default rules can be rendered inapplicable. Contracting information can never render default rules inapplicable due to interference with justifications or consequents, therefore when the process tree is being pruned one must only check the satisfaction of prerequisites. Of course removing information may mean that other previously blocked defaults can be used to generate the new extensions from the pruned process tree.

Procedure for contracting α:

1. Perform $W = W_\alpha^-$ using an appropriate contraction strategy.
2. For each branch in the process tree explicitly retract $W - W_\alpha^-$ from $In(\Pi(k))$, and then find the largest k such that $In(\Pi(k)) \vdash Pre(\delta_k)$.
3. Generate new extensions from the pruned process tree by checking all other defaults not used so far in a given process for success.

3.1.2 Denying Information

Denial is a stronger form of deleting α in which we prevent α from being included in any extension of the updated default theory. A natural way to ensure that contracted information, say α, is not reintroduced via the application of a default rule whilst generating the new extensions is to add its negation to $Out(\Pi)$. This can clearly be achieved by adding $\neg\alpha$ to all default justifications. *Exten* can represent defaults with multiple justifications. Formally: Given a default theory $T = (W, D)$ and a formula α, we define $T\text{-}\alpha = (W_\alpha^-, \{\frac{\varphi:\psi_1,\ldots,\psi_n,\neg\alpha}{\chi} \mid \frac{\varphi:\psi_1,\ldots,\psi_n}{\chi} \in D\})$.

An alternative less appealing approach would be to add a new default rule $\frac{true:\neg\alpha}{true}$ and either insist that it be used to generate all extensions using any number of techniques. For variants of default logic in which processes may fail $\frac{\alpha:true}{false}$[3] can be used. Constrained default logic is an example of a default logic where processes cannot fail.

We adopt the first solution which is more in the spirit of standard default logic.

Procedure for denying α:

1. Perform $W = W_\alpha^-$ using an appropriate contraction strategy.
2. Add $\neg\alpha$ to every default's set of justifications.
3. For each branch in the process tree explicitly retract $W - W_\alpha^-$ from $In(\Pi(k))$, and find the largest k such that $In(\Pi(k)) \vdash Pre(\delta_k)$.
4. Generate new extensions from the pruned process tree by checking all other defaults not used so far in a given process for success.

Example:

Consider the default theory $T = (W, D)$ with facts

a, b

and defaults

$$\delta_1 = \frac{true : p}{p}$$
$$\delta_2 = \frac{true\neg p}{\neg p}$$
$$\delta_3 = \frac{p : q}{q}$$
$$\delta_4 = \frac{q \wedge b : a \to r}{a \to r}$$
$$\delta_5 = \frac{true : \neg b}{\neg b}.$$

T has two extensions:

- $E_1 = Th(\{a, b, p, q, r\})$ which is generated, among others, by the process $\Pi_1 = (\delta_1, \delta_3, \delta_4)$[4].

[3]This default was suggested by an anonymous reviewer.

[4]*Exten* ensures that the process tree is pruned in a way that no two processes include exactly the same defaults.

- $E_2 = Th(\{a, b, \neg p\})$ which is generated by the process $\Pi_2 = (\delta_2)$.

Now suppose that we wish to deny b. First we remove b from the facts using an existent belief revision strategy. Then we turn to the two previous extensions E_i and the associated processes Π_i.

- Π_1 can be preserved up to the length of 2. Note that $Pre(\delta_4)$ is no longer derivable since b no longer exists. Next we check whether further defaults can be applied. In our case δ_5 is applicable and gives us the extension of the updated default theory $E_1' = Th(\{a, p, q, \neg b\})$. A part of this extension has been preserved from the previous computation of E_1.
- Π_2 can be fully preserved, but can be extended (step 4 in the procedure above) by the default δ_5 to give us the extension $E_2' = Th(\{a, \neg p, \neg b\})$.

3.1.3 Incorporating Information Consistent with the Extension

When we wish to add a formula α to a default theory $T = (W, D)$ such that $W \cup \{\alpha\}$ is consistent, we simply need to add α to W: $T + \alpha = (W \cup \{\alpha\}, D)$. The effect is that α is included in all extensions of the $T + \alpha$.

Procedure for expanding with α:

1. Perform $W = W \cup \{\alpha\}$.
2. For each successful branch in the process tree find the largest k such that $Th(In(\Pi(k)) \cup \{\alpha\}) \cap Out(\Pi(k)) = \emptyset$.
3. Generate new extensions from the pruned process tree by checking all other defaults not used so far in a given process for success.

3.1.4 Incorporating Information Inconsistent with the Extension

In this case it is not sufficient to just add α to W since this would lead to a default theory with one trivial (inconsistent) extension. Instead we need to revise W with α using a belief revision strategy: $T * \alpha = (W^-_{\neg\alpha} \cup \{\alpha\}, D)$. The effect is again that α is included in all extensions of the updated default theory.

Procedure for revising with α:

1. Perform $W = W^-_{\neg\alpha}$ using an appropriate contraction strategy.
2. For each successful branch in the process tree update $In(\Pi(k)) = Th((In(\Pi(k)) - \{W = W^-_{\neg\alpha}\}) \cup \{\alpha\})$, and find the largest k such that $In(\Pi(k)) \cap Out(\Pi(k) = \emptyset$ and $In(\Pi(k)) \vdash Pre(\delta_k)$.
3. Generate new extensions from the pruned process tree by checking all other defaults not used so far in a given process for success.

Example:

Consider the default theory $T = (W, D)$ with facts

a, b

and defaults

$$\delta_1 = \frac{true : p}{p}$$
$$\delta_2 = \frac{true : \neg p}{\neg p}$$
$$\delta_3 = \frac{p \wedge a : q}{q}$$
$$\delta_4 = \frac{\neg p : \neg q \wedge b}{\neg q}$$
$$\delta_5 = \frac{\neg b : r \wedge p}{r}.$$

T has two extensions:

- $E_1 = Th(\{a, b, p, q\})$ which is generated by the process $\Pi_1 = (\delta_1, \delta_3)$.
- $E_2 = Th(\{a, b, \neg p, \neg q\})$ which is generated by the process $\Pi_2 = (\delta_2, \delta_4)$.

Now suppose that we wish to revise by $\neg b$. First we remove b from the facts using an existent belief revision strategy, and add $\neg b$. Without loss of generality we assume a is retained in the revision process. Then we turn to the two previous extensions E_i and the associated processes Π_i.

- Π_1 can be preserved fully. Since δ_5 is applicable, Π_1 can be extended and leads to the extension of the updated default theory $E_1' = Th(\{a, \neg b, p, q, r\})$. A part of this extension has been preserved from the previous computation of E_1.
- Π_2 can be preserved up to length 1, since the justification of δ_4 is no longer consistent with the new $In(\Pi(1))$. The default δ_5 is not applicable, so the extension we obtain is $Th(\{a, \neg p, \neg b\})$.

3.2 CHANGING THE DEFAULT RULES

Addition and deletion of defaults are carried out as usual set operations on the set of defaults D. In the following we discuss how the extensions of the updated theory can be computed.

Procedure for adding a default rule δ:

1. For each closed and successful leaf in the process tree apply δ, and check for success.
2. Generate new extensions by checking all other defaults not used so far for success.

Procedure for removing a default rule δ_k from a successful process Π:

1. For each branch find $\Pi(k)$.
2. Generate new extensions from $\Pi(k)$ by checking all other defaults not used so far for success.

3.3 RESULTS

The strategies described in the previous section provide a sound basis upon which to build algorithms for changing a default theory. Theorem 1, below, shows that the strategy preserves the set of extensions.

Theorem 1: *Let $\mathcal{E}$ and $\mathcal{E}'$ be the set of extensions for the default theories (W, D) and (W', D'), respectively. The set of extensions generated for (W', D') via the strategy for pruning the process tree generated for (W, D) is precisely $\mathcal{E}'$.*

Theorem 2 below demonstrates that stratification can be used to identify the defaults that can be *ignored* after the pruning of the original process tree when generating the new extensions from the pruned tree.

Theorem 2: *Let $T = (W, D)$ be a default theory. Let ρ be a stratification function for T. If the least stratum in ρ of default rules pruned from a branch of a process tree for T is i then all default applications of default rules in strata strictly less than i are preserved by the pruning strategy.*

This result shows precisely how the performance of the generation of the new extensions from the pruned process tree can be improved using the stratification function. The stratification function remains in tact while ever the default rules are static, e.g. if the facts are changed. The stratification function must be recalculated when the default rules are added or removed.

Stratification plays a similar role in revising all the variants of default logic currently implemented in *Exten*, i.e. justified [15], constrained [10], and modified [15].

Finally, Theorem 3 below, says that the proposed strategy for pruning the process tree is minimal.

Theorem 3: *Let $\mathcal{P}$ and $\mathcal{P}'$ be the process trees for the default theories (W, D) and (W', D'), respectively. The process tree pruning strategy results in the largest subtree from the root in $\mathcal{P}$ contained in $\mathcal{P}'$.*

Theorem 3 establishes that given a process tree the proposed pruning process is the best one can hope for.

Pruning is dependent on the order of application of the default rules in building the original process tree. How drastically a process tree is pruned will depend somewhat on how late in the construction of the original process tree the effected default rules were applied. If stratification is used then this typically means that the process tree will be modified less if defaults that rely on the firing of numerous other defaults are the ones that are in conflict with the new information rather than defaults with weaker constraints, or with prerequisites satisfied by the facts.

4 DISCUSSION

One of the problems with default logic is that if the original default theory changes due to the acquisition of new information then not only must the new extensions be calculated from scratch, but default logic alone is incapable of adding new information that is inconsistent with the set of facts because it does not possess belief revision capabilities.

In this paper we provided a computational strategy for revising a default theory. The strategy is currently being used to augment an existing default logic system implemented in C^{++}, and to develop a new system in Java. Both systems build a process tree, the closed and successful leaves of which are extensions. The essential idea of the revision strategy is that it guides the system to the minimal pruning of a process tree. We established that the proposed pruning strategy preserves the set of extensions of the new default theory, and we demonstrated that stratification can be used to improve computational performance by identifying the defaults that can be *ignored* during the building the new extensions from the pruned process tree.

References

[1] Alchourrón, C., Gärdenfors, P., and Makinson, D., On the Logic of Theory Change: Partial Meet Functions for Contraction and Revision, *Journal of Symbolic Logic* 50: 510–530, 1985.

[2] G. Antoniou. *Nonmonotonic Reasoning.* MIT Press 1997.

[3] G. Antoniou, A. Courtney, J. Ernst and M.A. Williams. A System for Computing Constrained Default Logic Extensions. In *Proc. 5th European Workshop on Logics in AI (JELIA'96)*, Springer LNAI.

[4] G. Antoniou and M.A. Williams, *Reasoning with Incomplete and Changing Information*, Journal of Information Science, 99, 1 and 2: 83 - 99, 1997.

[5] P. Cholewinski. Stratified Default Logic. In *Proc. Computer Science Logic 1994*, Springer LNCS 933, 456-470.

[6] P. Cholewinski. Reasoning with Stratified Default Theories. In *Proc. 3rd International Conference on Logic Programming and Nonmonotonic Reasoning*, 1995.

[7] P. Cholewinski. Seminormal stratified default theories. *Annals of Mathematics and Artificial Intelligence* 1995.

[8] P. Cholewinski, W. Marek, A. Mikitiuk and M. Truszczynski. Experimenting with default logic. In *Proc. International Conference on Logic Programming*, MIT Press 1995.

[9] Courtney, A., Antoniou, G., and Foo, N.Y., Exten: A System for Computing Default Logic Extensions, *Proceedings of the 4th Pacific Rim International Conference on Artificial Intelligence*, LNAI 1114, Springer 1996, 471–482.

[10] J.P. Delgrande, T. Schaub and W.K. Jackson. Alternative approaches to default logic. *Artificial Intelligence* 70(1994): 167–237.

[11] Gärdenfors, P., and Makinson, D., *Revisions of Knowledge Systems using Epistemic Entrenchment*, Proceedings of the Second Conference on Theoretical Aspects of Reasoning about Knowledge, p83 – 96, 1988.

[12] G. Gottlob. Complexity results for nonmonotonic logics. *Journal of Logic and Computation* 2(1992): 397–425.

[13] Grove, A., *Two modellings for theory change*, Journal of Philosophical Logic, 17: 157 – 170, 1988.

[14] Lang, J., *Possibilistic Logic: Algorithms and Complexity* in J. Kohlas and S. Moral (eds), Handbook of Algorithms for Uncertainty and Defeasible Reasoning, Kluwer Academic Publishing.

[15] W. Lukaszewicz. Considerations on Default Logic. *Computational Intelligence* 4(1988): 1–16.

[16] MacNish, C.K. and Williams, M.A., *From Theory Revision to Design Revision: Applying Theory Change to Changing Requirements.* In G. Antoniou, A. Ghose and M. Truszczynski (Eds): *Learning and Reasoning with Complex Representations*, LNAI 1359, Springer, 207–222, 1998.

[17] W. Marek and M. Truszczynski. *Nonmonotonic Logic.* Springer 1993.

[18] A. Mikitiuk and M. Truszczynski. Constrained and rational default logics. In *Proc. International Joint Conference on Artificial Intelligence* 1995

[19] R. Reiter. A Logic for Default Reasoning. *Artificial Intelligence* 13(1980): 81–132.

[20] Spohn, W., *Ordinal Conditional Functions: A Dynamic Theory of Epistemic States*, In Harper, W.L., and Skyrms, B. (eds), Causation in decision, belief change, and statistics, II, Kluwer Academic Publishers, p105 – 134, 1988.

[21] H. Turner. Splitting a Default Theory. In *Proc. AAAI-96*, 645–651.

[22] Williams, M.A., *Transmutations of Knowledge Systems*, in J. Doyle, E. Sandewall, and P. Torasso (eds), Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference, Morgan Kaufmann, San Mateo, CA, 619 – 629, 1994.

[23] Williams, M.A., *Iterated Theory Base Change: A Computational Model*, in the Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, Montréal, 1541 – 1550, 1995.

[24] Williams, M.A., *Anytime Belief Revision*, In the Proceedings of the International Joint Conference on Artificial Intelligence, Morgan Kaufmann 1997, 74–79.

[25] Williams, M.A. and Foo, N.Y., *Nonmonotonic Dynamics of Default Logic*, in the Proceeding of the Ninth European Conference on Artificial Intelligence, Luigia Carlucci Aiello (ed), Pitman Publishing, London, 702 - 707, 1990.

[26] Williams, M.A., and Williams, D., *A Belief Revision Systems for the Wide World Web*, in the Proceedings of the IJCAI Workshop on Artificial Intelligence and the Internet, 1997.

Reasoning about Actions I

Anything Can Happen: on narratives and hypothetical reasoning

Lars Karlsson
Department of Computer and Information Science
Linköping University, S-581 83 Linköping, Sweden
Email: larka@ida.liu.se

Abstract

In this paper we consider the integration of three desirable properties in formalisms for action and change, namely representing and reasoning about (I) metric time, (II) alternative ways the world can develop relative to a specific choice of actions, and (III) alternative choices of actions. A language is presented that integrates these three aspects, and a translation to predicate calculus is provided. The central idea is to introduce narratives, that is chronological descriptions of what happens in the world over time, as objects which can be manipulated within the logic and to provide operators for reasoning about the properties of narratives.

1 INTRODUCTION

The purpose of this paper is to consider what it means to represent and reason about action structures such as sequences and narratives as objects in a logical language. Representing action structures as objects enables reasoning hypothetically about different choices of actions and their results. More concretely, this paper presents a logic, called the narrative logic (NL for short) for reasoning about action structures called narratives. The core of NL, that is the representation of actions and their effects, stems directly from Sandewall's PMON logic [24, 6]. From there, NL inherits a metric time structure and thereby provides a rich representation of temporal relations. On top of this PMON core, NL provides reasoning about different narratives, and the different ways the world can develop for a specific narrative. This is achieved by reifying two important concepts in PMON, namely the concept of an action schedule (that is narrative) and the concept of a development (that is a sequence of states over time). The combination of these features — explicit time, support for reasoning about alternative narratives and about alternative developments of a narrative — is what makes NL a novel and interesting formalism, suitable as a foundation for theories about plans and explanations.

When reasoning about action and change, branching time is one way to view time and actions and how they relate. In situation calculus [16, 22] (SC for short), the predominant event-based branching formalism, time is seen as generated by the execution of actions. There is an initial situation (denoted S_0), and the execution of an action in a situation results in a new situation, which yields a branching temporal structure (Fig. 1 (a)). A situation is denoted by a term constructed from the corresponding action sequence, e.g. $result(Fire, result(Load, S_0))$. The strength of this view is that it supports reasoning about alternative action sequences. In particular, action sequences are first-order objects and can for instance be quantified over. Thus, one can state such things as "if I load the gun and fire, the turkey will be dead": $\neg Holds(alive, result(Fire, result(Load, S_0)))$. Statements about situations in general are also possible. For instance, the fact that there is some sequence of actions (plan) that results in a situation where a condition (goal) G holds can be stated as $\exists s\,[G(s) \wedge ex(s)]$. Here, s is a situation variable and $ex(s)$ is a condition that the situation is reachable from the initial situation S_0. The weakness of SC is that there is no underlying, independent notion of time which actions can relate to. This makes it difficult to express nontrivial temporal properties of and relations between actions, such as partially ordered or overlapping action occurrences.[1]

[1]Although metric time can be added on top of the branching structure [20, 23], the underlying branching structure still enforces strong restrictions on how actions can be temporally related. In particular, the timing and ordering of actions needs to be complete.

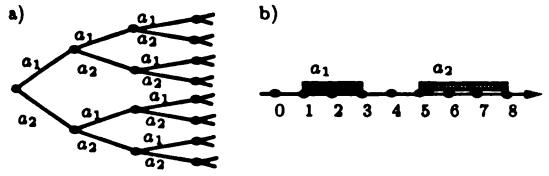


Figure 1: (a) Branching time structure formed by execution of actions where the nodes are situations; and (b) linear time structure and action occurrences.

A second way is to separate the notions of time and action, which is the case in narrative-based approaches [12, 2, 24]. Time is represented as a linear structure (for instance the integers), and the occurrence of an action is represented by connecting the action to the temporal structure with a special relation, like in $occurs(2, 5, Load)$ and $occurs(7, 9, Fire)$ (Fig. 1 (b)). The pros and cons of this approach are more or less complementary to the ones of situation calculus. Expressing temporal properties and relations is straightforward, but action structures (narratives) become meta-logical objects (logical statements), and reasoning about alternative narratives has to be done in terms of reasoning about alternative theories or logical statements. For instance, the statement "if I load the gun and fire, the turkey will be dead" has to be formulated $\Gamma \wedge \alpha \models \neg Holds(10, alive)$ or $\Gamma \models (\alpha \Rightarrow \neg Holds(10, alive))$ where α specifies what actions occur (*Load* and *Fire*) and Γ define action laws (among other things). Thus, it is to some extent possible to reason hypothetically about narratives, but when it comes to general statements such as the plan existence formulation mentioned in connection with SC above, then one has either to use second-order quantification or to go outside the logic and state that "there is a consistent α of the form X such that $\Gamma \wedge \alpha \models G$".

Besides notions of time and actions, reasoning about action and change usually involves a notion of state, that is what facts are true or false at a specific point in time. This is commonly represented with a *holds* relation between fluents (temporally dependent properties or relations) and situations or time-points. The states over a certain branch or line of time can be considered to form a development. In figure 1, each branch is associated with one (partial) development, and the narrative with one development.

However, given a specific action sequence or narrative, the world may develop in alternative ways due to different initial states and non-deterministic results of actions, and in the case of narratives, due to incomplete information regarding the timing and duration of actions. In many reasoning tasks it is important to be able to explicitly reason about these alternative developments of the world, for instance in plan synthesis (given observations of some initial state and a goal, find a sequence/narrative that necessarily leads to the goal) and explanation (given an observation of some state at a later time point, find a sequence/narrative that might have lead to that state). The keywords here are "necessarily" and "might". Neither the simple branching time nor linear time models presented above supports this form of reasoning, as there is only one development for each actions sequence/narrative.[2] Consequently, there are problems with the aforementioned SC formulation of plan existence. It has been shown that if one introduces incomplete initial information or nondeterministic effects of actions, then this formulation sometimes states that there is a plan when actually there is none [10]. The root of the problem is that it is not possible to formulate a condition that the plan should work for all potential initial states and nondeterministic results; instead, the formulation says "for each different combination of initial conditions/nondeterministic outcomes, there is a plan".

In order to deal with the reasoning problems mentioned above appropriately on the object level, one has to view actions sequences/narratives as entities having multiple parallel developments (Fig. 2). By quantifying over developments (either explicitly or with modal operators) one can distinguish between what must and what might be the result of a specific choice of actions. Dynamic logic [21] is a polymodal formalism with a branching temporal structure similar to the one of SC, which distinguishes between necessary and possible results of actions. Davis's [5] and Levesque's [15] versions of SC involve explicit theories of knowledge, and the knowledge operators there can be seen as "necessary"-operators. These two formalisms yield structures resembling Fig. 2 (a). Karlsson's SC version [10] does not have explicit modalities, but instead axiomatizes the existence of situations corresponding to each possible state in order to make both "necessary" and "possible" properties hold as intended. A noteworthy feature of the above-mentioned SC versions is that they introduce a type for action structures (sequences, conditionals etc) which is separate from situations, and reasoning about alternative action choices is done in terms of this type. Each action structure can be considered to be mapped to a set of developments. With a type for action structures, whether the underlying temporal structure is branching or linear

[2]Naturally, the different logical models of a situation calculus or narrative-based theory might represent different world developments. However, in each model there is only one such development, and syntactically it is not possible to construct statements that refer to multiple parallel developments.

should no longer be of importance for the ability to reason about different choices of actions.

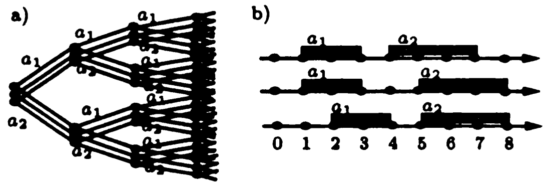


Figure 2: Branching (a) and linear (b) time structures with multiple parallel developments.

The narrative logic (NL), is a product of the insight that the key to reasoning about alternative choices of actions lies in the representation of action structures as objects and not in the underlying temporal structure, be it branching or linear. It is a formalism that has linear time and narratives as first-order objects associated with multiple developments. The fact that there is an independent notion of time enables a richer representation of temporal relations than is possible with event-based branching time. NL is based on the PMON entailment policy [24, 6] which has been proven correct for worlds with non-overlapping actions with context-dependent and non-deterministic effects. The central objects of NL are narratives, which have the following properties.

1. Narratives are terms in the logic, and can be quantified over and manipulated like other kinds of terms.
2. Narratives encode explicit assumptions or restrictions on what actions occur in the world.
3. These restrictions may be partial. For instance, the exact timing and duration of actions need not be specified.
4. These restrictions are local; typically, they concern specific intervals of time. Other types of locality, such as geographical locality, are possible but will not be considered in this paper.
5. Narratives can be incrementally extended to cover additional parts of the time line. Thus, narratives are dynamic entities, suitable for reasoning about alternative futures or pasts and for representing the beliefs of an agent in a dynamic environment.
6. Given a specific narrative, the world may develop in alternative ways due to incompleteness in the narrative (timing and duration of actions), different initial states and non-deterministic results of actions. Thus, a narrative corresponds to a set of developments, as in Fig. 2 (b).

The rest of the paper is organized as follows. Section 2 presents the syntax of NL and gives some examples. Section 3 shows how NL statements can be translated to predicate calculus and illustrates this with part of the examples from section 2. Section 4 presents some properties of narratives and and discusses applications, and section 5 presents some conclusions.

2 THE SURFACE LANGUAGE NL

In this section, the surface language NL is introduced. The different constructs of the language are presented together with their intuitions. In the next section, a translation from NL to predicate calculus (PC) is provided (where an induction axiom and four development existence axioms are second-order).

The basic types of NL are $\mathcal{F}$ for fluents, $\mathcal{A}$ for actions, $\mathcal{T}$ for time-points (non-negative integers) and $\mathcal{S}$ for temporal designators. The variable symbols f, a, t and s will be used for these types, respectively.

Narratives are constructed using narrative components that specify what actions occur, and temporal designators. The latter specify the time interval of the narrative component. Furthermore, they are used for temporal relations between actions. Notice that temporal designators are of a type $\mathcal{S}$ which is different from the time-point type $\mathcal{T}$. This is due to point 3 in the introduction; narratives are partial restrictions, and permit variations of the timing of actions. If time-points ($\mathcal{T}$) had been used, the timing of all actions would have been totally fixed, and thus identical in all developments of the narrative. To provide a possibility to refer to fixed time-points in a narrative, there is a function @ that turns a time-point into a temporal designator.

Definition 1 *A temporal designator $s \in \mathcal{S}$ is a term of the form* Sn, $n \in \mathbf{N}$ *(for unfixed designators),* @t, $t \in \mathcal{T}$ *(for fixed designators) or* $s_1 + s_2$ *(or alternatively to keep a term-like syntax:* PLUS(s_1, s_2)*). An ordering constraint $c \in \mathcal{C}$ is a term of the form* $(s = s')$, $(s \leq s')$, $(s < s')$, $(\neg c)$, $(c_1 \wedge c_2)$, $(c_1 \vee c_2)$ *or* T *for true (or alternatively:* EQ(s, s') *etc.).*

Definition 2 *A narrative component $n \in \mathcal{N}$ is either a variable n or a term of the form* D(s, s', a) *(single action) or* J$[n_1, \ldots, n_k, c]$ *where $k \geq 0$ (*J *stands for "join").*

The join operator combines subcomponents and relates them temporally (c is an ordering constraint).

Definition 3 *Let s, s' be temporal designators, f a fluent, $V \in \{\mathrm{T}, \mathrm{F}\}$ and a an action. A holds statement* *is of the form* H(s, f)*, a do statement is of the form* D(s, s', a)*, and a reassignment statement is of the form* R(s, s', f, V)*. A temporal statement is of the form* $s = s'$, $s \leq s'$ *or* $s < s'$.

The H construct refers to the truth value of a particular fluent at a particular time point, and D to the occurrence of a particular action. The R construct denotes that the value of a fluent f is reassigned to the truth value V during the given interval $(s, s']$. It is used exclusively for specifying the effects of actions. Notice the overloading of the D symbol and the temporal relations. They are used both for terms and statements; however, it should always be clear from the context what is intended.

A development is characterized by what actions occur and what fluents hold over time, and how temporal designators are mapped to time-points. The Law construct is used for stating axioms regarding the effects of actions on fluents. These, together with the principle of persistence (fluents only change due to the effects of actions) and a restriction that action occurrences may not temporally overlap, determine the set of (well-behaved) developments.

Definition 4 *A law statement is of the form*

$$\text{Law}(\ \forall s, s'.\text{D}(s, s', a) \Rightarrow \bigwedge_i[\alpha_i^s \Rightarrow \bigvee_j(\bigwedge_k \text{R}(s, s', f_{ik}, V_{ijk})) \wedge \beta_{ijk}^{s,s'})]\ ) \quad (1)$$

where α_i^s *is a boolean combination of holds statements at time* s *and* $\beta_{ijk}^{s,s'}$ *(optional) is a logical combination of temporal statements and holds statements referring to fluents* f_{ik} *which specifies what holds in the interval* (s, s') *and the length of this interval.*

Example: The following are laws for a loading and a firing action.

$$\text{Law}(\ \forall s, s'.\text{D}(s,s',\text{LOAD}) \Rightarrow \text{R}(s,s',\text{LOADED},\text{T})\ ) \quad (2)$$

$$\text{Law}(\ \forall s, s'.\text{D}(s,s',\text{FIRE}) \Rightarrow (\ \text{H}(s,\text{LOADED}) \Rightarrow \text{R}(s,s',\text{LOADED},\text{F}) \wedge \text{R}(s,s',\text{ALIVE},\text{F})\ )\ ) \quad (3)$$

These general laws are then used for reasoning about specific narratives. Relative to a narrative, the world may develop in many different ways (see point 6 in the introduction). Therefore, NL has two operators of a modal character.

Definition 5 *A narrative consequence statement is of the form* Nec(s, s', n, α) *(necessary consequence) or* Pos(s, s', n, α) *(possible consequence) where* s, s' *are temporal designators,* n *is a narrative component, and* α *is a logical combination of holds, do, temporal, and narrative consequence statements.*

Essentially, Nec(s, s', n, α) states that for each development where the set of actions occurring within $[s, s']$ are exactly those in n and nothing is assumed about what occurs outside $[s, s']$, it is the case that α holds. That is to say, α is an inevitable consequence of the action occurrences in the narrative component n. A formal definition is given in section 3.2.2. Pos(s, s', n, α) is defined as $\neg$Nec($s, s', n, \neg\alpha$).

Example: Given the laws (2), (3) above, the statements (4), (5) below are true. They state that if the gun is first loaded and then fired and no other action occurs in the interval [@0, S4], then it is a necessary consequence that the unfortunate turkey that is the target will die ($\neg$H(S4, ALIVE) holds in all developments), but if only the fire action occurs, then it is possible that the turkey will survive (H(S3, ALIVE) holds in some developments).

$$\text{Nec}(\ @0,\ \text{S4},\ \text{J}[\text{D}(\text{S1},\text{S2},\text{LOAD}),\text{D}(\text{S3},\text{S4},\text{FIRE}), (@0<\text{S1})\wedge(\text{S2}\leq\text{S3})],\ \neg\text{H}(\text{S4},\ \text{ALIVE})\ ) \quad (4)$$

$$\text{Pos}(\ @0,\ \text{S3},\ \text{J}[\text{D}(\text{S1},\text{S2},\text{FIRE}), (@0<\text{S1})\wedge(\text{S2}\leq\text{S3})],\ \text{H}(\text{S3},\ \text{ALIVE})\ ) \quad (5)$$

Informally, the truth of these statements can be justified as follows. Regarding (4), D(S1,S2,LOAD) and (2) imply H(S2,LOADED), and the principle of persistence implies H(S3,LOADED). Then, D(S3,S4,FIRE) and (3) imply $\neg$H(S4,ALIVE). Regarding (5), assume a development were $\neg$H(@0,LOADED) $\wedge$ H(@0,ALIVE) is true. By persistence, $\neg$H(S1,LOADED) is the case, and therefore D(S1,S2,FIRE) has no effects according to (3). So, by persistence H(S3,ALIVE) holds in this particular development. Therefore, H(S3,ALIVE) is a possible consequence.

Nec and Pos operators can also be nested, and nesting implies a gradual extension of a narrative (point 4 in the introduction). To make nesting of Nec and Pos operators meaningful, we need to refine the semantics of the Nec operator given above. For that purpose, we introduce the concept of a partial development, that is an equivalence class of developments that are equal within some given time segments. As the restrictions imposed by a narrative component on developments only concern a segment of time and nothing is assumed outside this segment, a narrative component can be seen as specifying a set of partial developments that are defined only over the associated time segment. Further, if the narrative component is applied to an existing partial development, then the narrative component can be considered to extend that partial development to also include the associated temporal segment in a way that satisfies the restrictions encoded in the narrative component. Often, a partial development can be extended in more than one way, so an extension results in a set of new partial developments.

Consequently, nesting of Nec and Pos operators implies a gradual extension of a set of partial developments. One starts with a set containing only the empty partial development (that is not defined for any temporal segment), then applies the narrative of the outermost operator which will result in a set of partial developments defined for the time segment of the outermost operator, and then applies the next operator to that set, and so on.

Based on the concept of a partial development, we can now give the following (informal) semantics for the Nec operator applied in the context of a partial development. *The expression* Nec(s, s', n, α) *states that for each new partial development that can be obtained by extending the given partial development over* $[s, s']$ *in such a way that the set of actions occurring within* $[s, s']$ *are exactly those in* n, *it is the case that the statement* α *holds.* For Pos, just replace "for each" with "for some" above.

Example: Consider the following statement.

$$\begin{array}{l}\text{Nec}(\ @0,\ \text{S2},\ \text{J}[\text{D}(\text{S1},\text{S2},\text{LOAD}),\ (@0\leq \text{S1})],\\ \quad \text{Nec}(\ \text{S2},\ \text{S4},\ \text{J}[\text{D}(\text{S3},\text{S4},\text{FIRE}),(\text{S2}\leq \text{S3})],\\ \qquad \neg\text{H}(\text{S4},\ \text{ALIVE})\ )\ )\end{array} \tag{6}$$

Eq. 6 states that whenever the loading action is executed, proceeding by executing the firing action inevitably results in the turkey's demise. That is to say, any partial development where the gun is loaded can only be extended with the gun being fired in such a way that the turkey dies. Notice the structure of the statement: the last argument of the outer Nec consists of the inner Nec statement, which in turn has $\neg$H(S4, ALIVE) as its last argument.

Informally, the truth of (6) can be justified as follows. Take any partial development of the outermost narrative. The only action occurrence in the temporal segment [@0,S2] is D(S1,S2,LOAD). From D(S1,S2,LOAD) and (2) follows H(S2,LOADED). Now extend that partial development according to the inner narrative. As H(S2,LOADED) already holds, it follows from persistence that H(S3,LOADED) is true. Then, D(S3,S4,FIRE) and (3) imply $\neg$H(S4,ALIVE).

The Nec and Pos operators can be mixed with each other, and logically combined with holds, do and temporal statements. Finally, consistency of a narrative, i.e. the narrative corresponds to some possible development of the world, can be stated as Pos(s, s', n, T).[3]

[3]Notice that narrative consistency is a property of objects in an NL theory, and not of the NL theory itself. Yet, there is a strong analogy: logical consistency means that a theory has models, and narrative consistency means that a narrative has developments.

Observe that there is a large degree of freedom in the temporal relations of a narrative component. It is easy to construct narrative consequence statements Nec(s, s', n, α) where actions in the narrative component n are not constrained to occur inside the given time segment $[s, s']$. The Within operator characterizes those components that actually do have their actions within the assigned time segment.

Definition 6 *A narrative verification statement is of the form* Within(s, s', n) *(all actions in* n *start and end within the interval defined by* s, s'*).*

The concept of a theory is defined as follows.

Definition 7 *An NL theory* Γ_L *is a finite set of law statements.*

Finally, $\models_{NL}$ denotes the consequence relation between an NL theory and a logical combination of narrative consequence (Nec and Pos) and verification statements α. The definition of $\models_{NL}$ is presented in section 3, and is based on a translation from NL to predicate calculus. In section 4, we return to properties and applications of the formalism.

3 A TRANSLATION TO PC

In this section, a translation from NL to second-order predicate calculus is outlined. This implies that both a proof theory and a semantics are indirectly provided. The concept of a development, that is to say a precise description of what happens and holds in the world over time, plays a central role. The elements of this translation are as follows.

1. Narratives and developments are defined as first-order objects in PC.
2. A specification of the set of well-behaved developments is given in PC. A well-behaved development is one that obeys the action laws and the principle of persistence.
3. Next, the relation between narratives and developments is specified in PC. The concept of a narrative describing a development is defined.
4. A function for translating narrative logic statements to PC statements is defined.
5. Finally, a consequence relation is defined for NL based on this translation.

Points 1-3 constitute the base theory Γ_B of NL, and to this the translation of an NL theory Γ_L is added (point 4).

3.1 BASE THEORY

The base theory Γ_B consists of the axioms presented below, and unique names and (optionally) domain closure axioms.

3.1.1 Narrative

A narrative is basically a sequence of interval- and narrative component pairs.

Definition 8 *A narrative $\boldsymbol{m} \in \mathcal{M}$ is a term of the form ϵ (for the empty narrative) or* $\mathrm{E}(\boldsymbol{m}, \boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{n})$ *(the narrative $\boldsymbol{m}$ extended with the component $\boldsymbol{n}$ over the interval $[\boldsymbol{s}_1, \boldsymbol{s}_2]$).*

There is an induction axiom over narrative components (we utilize the fact that any join $\mathrm{J}[\boldsymbol{n}_1, \ldots, \boldsymbol{n}_n, \boldsymbol{c}]$ can be rewritten $\mathrm{J}[\boldsymbol{n}_1,\mathrm{J}[\ldots,\mathrm{T}],\mathrm{c}]$). $\mathrm{J}[\mathrm{T}]$ is the empty narrative component.

$$\begin{aligned}&\forall P[(\forall s, s', a[P(\mathrm{D}(s, s', a))] \wedge P(\mathrm{J}[\mathrm{T}]) \wedge \\ &\quad \forall n, n', c[P(n) \wedge P(n') \Rightarrow P(\mathrm{J}[n, n', \mathrm{c}])]) \Rightarrow \\ &\quad \forall n[P(n)]]\end{aligned} \tag{7}$$

3.1.2 Developments

Developments specify what actions occur and what fluents are true or false over time. Developments have the status of composite first-order objects in the PC translation, and the four components of a development define what actions occur, what fluents are true, what fluents might change, and how temporal designators are mapped to the time line in that particular development.

Definition 9 *A development $\delta \in \Delta$ is a tuple $\langle \boldsymbol{d}, \boldsymbol{h}, \boldsymbol{o}, \boldsymbol{v}\rangle \in (\mathcal{D} \times \mathcal{H} \times \mathcal{O} \times \mathcal{V})$ where*

- *the relation $D(\boldsymbol{d}, \boldsymbol{t}, \boldsymbol{t}', \boldsymbol{a})$ characterizes what actions occur;*
- *the relation $H(\boldsymbol{h}, \boldsymbol{t}, \boldsymbol{f})$ characterizes what fluents hold;*
- *the relation $X(\boldsymbol{o}, \boldsymbol{t}, \boldsymbol{f})$ characterizes what fluents might change due to reassignment;*
- *the function $V(\boldsymbol{v}, \boldsymbol{s})$ characterizes the values of designators $\boldsymbol{s}$. The co-domain of V is $\mathcal{T}$.*

When referring to the individual $\boldsymbol{d}$, $\boldsymbol{h}$, $\boldsymbol{o}$ and $\boldsymbol{v}$ components of a development δ, the notation δ^d, δ^h, δ^o and δ^v is used. Thus, the fact that an action occurs in the development δ is written $D(\delta^d, t, t', a)$, and that a fluent holds and is occluded is written $H(\delta^h, t, f)$ and $X(\delta^o, t, f)$, respectively. Finally, the value of a temporal designator in δ is obtained from $V(\delta^v, s)$.

Four second-order development existence axioms define the set of developments.[4] They state that for each possible instantiation Φ of the D predicate, there is a corresponding $\boldsymbol{d} \in \mathcal{D}$ and so on for H, X and V.

$$\forall \Phi \exists d \forall t, t', a[\Phi(t, t'a) \equiv D(d, t, t', a)] \tag{8}$$

$$\forall \Phi \exists h \forall t, f[\Phi(t, f) \equiv H(h, t, f)] \tag{9}$$

$$\forall \Phi \exists o \forall t, f[\Phi(t, f) \equiv X(o, t, f)] \tag{10}$$

$$\begin{aligned}\forall \Phi[\ &(\forall t[\Phi(@(t)) = t] \wedge \\ &\forall s, s'[\Phi(s + s') = \Phi(s) + \Phi(s')]) \equiv \\ &\exists v[\forall s[\Phi(v, s) = V(v, s)]]\end{aligned} \tag{11}$$

Observe that (11) implies that $\forall v, t[V(v, @(t)) = t]$, that is to say @ has the intended meaning. [5]

3.1.3 Well-behaved developments

Each development tuple $\delta = \langle \boldsymbol{d}, \boldsymbol{h}, \boldsymbol{o}, \boldsymbol{v}\rangle$ represents a specific instantiation of the H, D and X relations and the V function. However, only a subset of these combinations represent well-behaved developments; namely those that satisfy the action laws and the principle of persistence. The predicate Wb determines this subset. Wb is a first-order version of the PMON entailment policy [24, 6]. In short, Wb states that a development should (a) satisfy the action laws (Law is generated from the Law statements; see 3.2.4), (b) have all action occurrences sequentially ordered, (c) have a minimal extension of occlusion (X), and (d) change may only occur at occluded timepoint-fluent pairs ($\oplus$ is exclusive or). It is points (c) and (d) that formalize the principle of persistence. X occurs in the translation of reassignment statements which are used for the effects of actions, and points (c) and (d) simply restrict the change of fluents to those cases when the fluent actually is influenced by an action. We assume the

[4]A parallel can be found in Baker's second-order situation existence axiom [3]. This axiom, used for obtaining a correct minimization of change, states that for each possible truth assignment for fluents, there is a situation where *Holds* realizes this assignment.

[5]In addition, notice that the relation/function associated with a specific development component can be encoded as a set of pairs of natural numbers if the sets of actions, fluents and designators are countable. Thus, the cardinalities of the different sets of (equivalence classes of) development components are $2^{\aleph_0}$, the same as for the real numbers.

convention that free variables (such as δ below) are always universally quantified over.

$$
\begin{aligned}
& Wb(\delta) \equiv \{ \\
& \text{(a) } Law(\delta) \wedge \\
& \text{(b) } Seq(\delta^d) \wedge \\
& \text{(c) } \neg \exists o' \; [\forall t, f[X(o', t, f) \Rightarrow X(\delta^o, t, f)] \wedge \\
& \qquad \neg \forall t, f[X(\delta^o, t, f) \Rightarrow X(o', t, f)] \wedge \\
& \qquad Law(\langle \delta^d, \delta^h, o', \delta^v \rangle)\,] \wedge \\
& \text{(d) } \forall f, t[\; H(\delta^h, t, f) \oplus H(\delta^h, t+1, f) \Rightarrow \\
& \qquad X(\delta^o, t+1, f)]\}
\end{aligned}
\tag{12}
$$

The sequentiality condition is defined as follows.

$$
\begin{aligned}
& Seq(d) \equiv \\
& \quad \forall t_1, t_1', a_1, t_2, t_2', a_2[\\
& \quad\; (D(d, t_1, t_1', a_1) \Rightarrow t_1 < t_1') \wedge \\
& \quad\; (D(d, t_1, t_1', a_1) \wedge D(d, t_2, t_2', a_2) \Rightarrow \\
& \quad\;\; (t_1' \le t_2 \vee t_2' \le t_1 \vee \\
& \quad\;\;\; \langle t_1, t_1', a_1 \rangle = \langle t_2, t_2', a_2 \rangle))\,]
\end{aligned}
\tag{13}
$$

3.1.4 Relating narratives to developments

A narrative component describes a set of developments, namely those well-behaved developments that contain the same actions as the narrative component in the associated time segment. A predicate *De* formalizes this relation. It requires that all ordering constraints in the narrative component hold (Cn) and that actions occur in the assigned interval if and only if they are present in the narrative component (In).

$$
\begin{aligned}
& De(s, s', n, \delta) \equiv \\
& \{Cn(n, \delta) \wedge V(\delta^v, s) \le V(\delta^v, s') \wedge \\
& \forall t, t', a[\; (V(\delta^v, s) \le t < V(\delta^v, s') \vee \\
& \qquad V(\delta^v, s) < t' \le V(\delta^v, s')) \Rightarrow \\
& \qquad (D(\delta^d, t, t', a) \equiv In(\delta, t, t', a, n))]\,\}
\end{aligned}
\tag{14}
$$

The In predicate formalizes the presence of an action inside a narrative component. Notice that the action a is associated with two time-points t, t' which are matched against the temporal designators in the narrative component.

$$
\begin{aligned}
& In(\delta, t, t', a, \mathrm{D}(s, s', a)) \equiv \\
& \quad (\langle t, t', a \rangle = \langle V(\delta^v, s), V(\delta^v, s'), a \rangle)
\end{aligned}
\tag{15}
$$

$$
\begin{aligned}
& In(\delta, t, t', a, \mathrm{J}[n_1, \ldots, n_k, c]) \equiv \\
& \quad (\textstyle\bigvee_i In(\delta, t, t', a, n_i))
\end{aligned}
\tag{16}
$$

The Cn predicates state that the temporal constraints specified in a narrative component/ordering constraint hold. Let $\mathbf{C} \in \{\wedge, \vee\}$ and $\mathbf{R} \in \{=, \le, <\}$.

$$
\begin{aligned}
& Cn(\delta, \mathrm{J}[n_1, \ldots, n_k, c]) \equiv \\
& \qquad (\wedge_i Cn(\delta, n_i)) \wedge Cn(\delta, c)
\end{aligned}
\tag{17}
$$

$$Cn(\delta, \mathrm{D}(s, s', a)) \equiv [V(\delta^v, s) < V(\delta^v, s')] \tag{18}$$

$$Cn(\delta, \mathrm{T}) \equiv T \tag{19}$$

$$Cn(\delta, (\neg c)) \equiv [\neg Cn(\delta, c)] \tag{20}$$

$$Cn(\delta, (c_1 \mathbf{C} c_2)) \equiv [Cn(\delta, c_1) \mathbf{C} Cn(\delta, c_2)] \tag{21}$$

$$Cn(\delta, (s_1 \mathbf{R} s_2)) \equiv [V(\delta^v, s_1) \mathbf{R} V(\delta^v, s_2)] \tag{22}$$

Example: The narrative component in (4) yields the following:

$$
\begin{aligned}
& De(@0,\ \mathrm{s4},\ \mathrm{J}[\mathrm{D(s1,s2,LOAD)},\ \mathrm{D(s3,s4,FIRE)}, \\
& \qquad (@0{<}\mathrm{s1}) \wedge (\mathrm{s2} \le \mathrm{s3})],\ \delta) \quad \equiv \\
& \quad \{\, 0 \le V(\delta^v, \mathrm{s4}) \wedge \\
& \quad\; (0 < V(\delta^v, \mathrm{s1}) \wedge V(\delta^v, \mathrm{s2}) \le V(\delta^v, \mathrm{s3}) \wedge \\
& \quad\; V(\delta^v, \mathrm{s1}) < V(\delta^v, \mathrm{s2}) \wedge \\
& \quad\; V(\delta^v, \mathrm{s3}) < V(\delta^v, \mathrm{s4})) \wedge \\
& \quad\; \forall t, t', a[(\; 0 \le t < V(\delta^v, \mathrm{s4}) \vee \\
& \quad\;\; V(\delta^v, 0) < t' \le V(\delta^v, \mathrm{s4})) \Rightarrow \\
& \quad\; [D(\delta^d, t, t', a) \equiv \\
& \quad\;\; (\langle t, t', a \rangle = \langle V(\delta^v, \mathrm{s1}), V(\delta^v, \mathrm{s2}), \mathrm{LOAD} \rangle \vee \\
& \quad\;\;\; \langle t, t', a \rangle = \langle V(\delta^v, \mathrm{s3}), V(\delta^v, \mathrm{s4}), \mathrm{FIRE} \rangle)\,]\,]\,\}
\end{aligned}
\tag{23}
$$

Finally, there is a predicate Eq that represents that two developments are equal in the time intervals covered by a narrative. Recall the discussion about partial developments in section 2. A partial development is an equivalence class of complete developments that are equal within some temporal segments. Eq defines when two complete developments belong to the same equivalence class given a specific narrative.

$$Eq(\delta_1, \delta_2, \epsilon) \tag{24}$$

$$
\begin{aligned}
& Eq(\delta_1, \delta_2, \mathrm{E}(m, s, s', n)) \equiv \{ \\
& \; V(\delta_1^v, s) = V(\delta_2^v, s) \wedge V(\delta_1^v, s') = V(\delta_2^v, s') \wedge \\
& \; Eq(\delta_1, \delta_2, V(\delta_1^v, s), V(\delta_1^v, s')) \wedge \\
& \; Eq(\delta_1, \delta_2, m)\,\}
\end{aligned}
\tag{25}
$$

$$
\begin{aligned}
& Eq(\delta_1, \delta_2, t_1, t_2) \equiv \{ \\
& \; \forall t, t', a[t_1 \le t < t' \le t_2 \Rightarrow \\
& \; (\, D(\delta_1^d, t, t', a) \equiv D(\delta_2^d, t, t', a) \wedge \\
& \quad \exists t'' D(\delta_1^d, t, t'', a) \equiv \exists t'' D(\delta_2^d, t, t'', a)) \wedge \\
& \quad \exists t'' D(\delta_1^d, t'', t', a) \equiv \exists t'' D(\delta_2^d, t'', t', a)\,)\,] \wedge \\
& \; \forall t'[\; t_1 \le t' \le t_2 \Rightarrow \\
& \qquad \forall f[H(\delta_1^h, t', f) \equiv H(\delta_2^h, t', f)] \wedge \\
& \qquad \forall f[X(\delta_1^o, t', f) \equiv X(\delta_2^o, t', f)] \wedge \\
& \qquad \forall s[V(\delta_1^v, s) = t' \equiv V(\delta_2^v, s) = t']\,]\,\}
\end{aligned}
\tag{26}
$$

3.2 TRANSLATION FUNCTION

This subsection presents the translation $[\cdot]_m^\delta$ from $\mathcal{NL}$ statements to predicate logic statements. The superscript δ and subscript m are function arguments that refer to the development and narrative expressions that form the context of the translation. The logical connectives and quantifiers are translated in the obvious way. Whenever a new variable is introduced, it is assumed to be fresh.

3.2.1 Basic statements

The translation from basic NL statements to predicate logic statements is straight-forward: H to H, D to D, R to H and X, and temporal designators to V, as follows ($\mathbf{R} \in \{=, \leq, <\}$).

$$[\boldsymbol{s}_1 \mathbf{R} \boldsymbol{s}_2]^{\delta}_{m} = V(\delta, \boldsymbol{s}_1)\mathbf{R}V(\delta, \boldsymbol{s}_2) \tag{27}$$

$$[\mathrm{H}(\boldsymbol{s}, \boldsymbol{f})]^{\delta}_{m} = H(\delta^h, V(\delta^v, \boldsymbol{s}), \boldsymbol{f}) \tag{28}$$

$$[\mathrm{D}(\boldsymbol{s}, \boldsymbol{s}', \boldsymbol{a})]^{\delta}_{m} = D(\delta^d, V(\delta^v, \boldsymbol{s}), V(\delta^v, \boldsymbol{s}'), \boldsymbol{a}) \tag{29}$$

$$\begin{aligned}&[\mathrm{R}(\boldsymbol{s}, \boldsymbol{s}', \boldsymbol{f}, \mathrm{T})]^{\delta}_{m} = \\ &\quad (H(\delta^h, V(\delta^v, \boldsymbol{s}'), \boldsymbol{f}) \wedge \\ &\quad \forall t[V(\delta^v, \boldsymbol{s}) < t \leq V(\delta^v, \boldsymbol{s}') \Rightarrow X(\delta^o, t, \boldsymbol{f})])\end{aligned} \tag{30}$$

$$\begin{aligned}&[\mathrm{R}(\boldsymbol{s}, \boldsymbol{s}', \boldsymbol{f}, \mathrm{F})]^{\delta}_{m} = \\ &\quad (\neg H(\delta^h, V(\delta^v, \boldsymbol{s}'), \boldsymbol{f}) \wedge \\ &\quad \forall t[V(\delta^v, \boldsymbol{s}) < t \leq V(\delta^v, \boldsymbol{s}') \Rightarrow X(\delta^o, t, \boldsymbol{f})])\end{aligned} \tag{31}$$

The translation of R is of special interest. Reassignment implies that the fluent $\boldsymbol{f}$ is true/false at the end of the interval $(\boldsymbol{s}, \boldsymbol{s}']$, *and* that $\boldsymbol{f}$ is not subject to persistence within the interval $(\boldsymbol{s}, \boldsymbol{s}']$.

3.2.2 Narrative consequence statements

The translation of the Nec operator is as follows; Pos($\boldsymbol{s}, \boldsymbol{s}', \boldsymbol{n}, \alpha$) is defined as $\neg$Nec($\boldsymbol{s}, \boldsymbol{s}', \boldsymbol{n}, \neg\alpha$).

$$\begin{aligned}&[\mathsf{Nec}(\ \boldsymbol{s}, \boldsymbol{s}', \boldsymbol{n}, \alpha\ )]^{\delta}_{m} = \\ &\quad \forall\delta'[(Eq(\delta, \delta', m) \wedge Wb(\delta') \wedge De(\boldsymbol{s}, \boldsymbol{s}', \boldsymbol{n}, \delta')) \Rightarrow \\ &\qquad [\alpha]^{\delta'}_{\mathrm{E}(m, \boldsymbol{s}, \boldsymbol{s}', \boldsymbol{n})}]\end{aligned} \tag{32}$$

This definition states that for each development δ' such that (a) δ' is equal to δ in the temporal segments in m, (b) δ' is well-behaved, and (c) the set of actions occurring within $[s, s']$ in δ' are exactly those in n and the ordering constraints in n are satisfied, the translation of α in the context of δ' and $\mathrm{E}(m, s, s', n)$ must hold. Compare this to the informal definition of Nec that was given in section 2. Condition (a) above corresponds to the condition in the informal definition that the new partial development should be an extension of the given partial development.

Example: The translation of (4) is as follows, where $\boldsymbol{n}$ is the narrative component in question.

$$\begin{aligned}&\forall\delta, \delta_2[Eq(\delta, \delta_2, \epsilon) \wedge Wb(\delta_2) \wedge De(@0, \mathrm{s}4, \boldsymbol{n}, \delta_2) \Rightarrow \\ &\qquad \neg H(\delta^h_2, V(\delta^v_2, \mathrm{s}4), \mathrm{ALIVE})]\end{aligned} \tag{33}$$

3.2.3 Narrative verification statements

The Within operator is defined as follows.

$$\begin{aligned}&[\mathsf{Within}(\boldsymbol{s}, \boldsymbol{s}', \boldsymbol{n})]^{\delta}_{m} = \\ &\quad \forall\delta_2[(Eq(\delta, \delta_2, m) \wedge Wb(\delta_2) \wedge Cn(\delta_2, \boldsymbol{n})) \Rightarrow \\ &\qquad \forall t, t', a[\ In(\delta_2, t, t', a, \boldsymbol{n}) \Rightarrow \\ &\qquad\qquad V(\delta^v_2, \boldsymbol{s}) \leq t < t' \leq V(\delta^v_2, \boldsymbol{s}')]]\end{aligned} \tag{34}$$

3.2.4 Law statements and NL theories

For an NL theory Γ_L = {Law(α_1),..., Law(α_k)}, the translation function is defined as follows:

$$[\Gamma_L]^{\delta}_{m} = (Law(\delta) \equiv (\textstyle\bigwedge^{k}_{i=1}[\alpha_i]^{\delta}_{m})) \tag{35}$$

Example: The laws in (2), (3) yield the following definition of *Law*:

$$\begin{aligned}&Law(\delta) \equiv \forall s, s'[\\ &\quad D(\delta^d, V(\delta^v, s), V(\delta^v, s'), \mathrm{LOAD}) \Rightarrow \\ &\qquad (H(\delta^h, V(\delta^v, s'), \mathrm{LOADED}) \wedge \\ &\qquad \forall t[V(\delta^v, s) < t \leq V(\delta^v, s') \Rightarrow \\ &\qquad\quad X(\delta^o, t, \mathrm{LOADED}))] \wedge \\ &\quad D(\delta^d, V(\delta^v, s), V(\delta^v, s'), \mathrm{FIRE}) \Rightarrow \\ &\qquad H(\delta^h, V(\delta^v, s), \mathrm{LOADED}) \Rightarrow \\ &\qquad (\neg H(\delta^h, V(\delta^v, s'), \mathrm{ALIVE}) \wedge \\ &\qquad \forall t[V(\delta^v, s) < t \leq V(\delta^v, s') \Rightarrow \\ &\qquad\quad X(\delta^o, t, \mathrm{ALIVE})] \wedge \\ &\qquad \neg H(\delta^h, V(\delta^v, s'), \mathrm{LOADED}) \wedge \\ &\qquad \forall t[V(\delta^v, s) < t \leq V(\delta^v, s') \Rightarrow \\ &\qquad\quad X(\delta^o, t, \mathrm{LOADED})]\,)\,]\end{aligned} \tag{36}$$

The following theorem makes it possible to obtain a more convenient formulation of what is occluded than the statement about minimal occlusion provided in the definition of Wb (12(c)).

Theorem 1 *Let $\Gamma = \Gamma_B \wedge [\Gamma_L]$ be a translated theory. Then the definition of Wb (12) is logically equivalent to a sentence*

$$\begin{aligned}&Wb(\delta) \equiv \{ \\ &\ (\mathrm{a})\ Law(\delta) \wedge \\ &\ (\mathrm{b})\ Seq(\delta^d) \wedge \\ &\ (\mathrm{c})\ \forall f, t[X(\delta^o, t, f) \equiv \Lambda(\delta, t, f)] \wedge \\ &\ (\mathrm{d})\ \forall f, t[\ H(\delta^h, t, f) \oplus H(\delta^h, t{+}1, f) \Rightarrow \\ &\qquad X(\delta^o, t{+}1, f)]\}\end{aligned} \tag{37}$$

where $\Lambda(\delta, t, f)$ is defined entirely in terms of D, H and comparisons between temporal, fluent and action terms.

Proof (sketch). In action laws, R only occurs positively (1), and due to the translation of R (30), (31), X will only occur positively inside the definition of *Law* that results from the translation (35). Thus, it is possible to reformulate (35) after translation as $\forall\delta, m[Law(\delta) \equiv (\forall f, t[\Lambda(\delta, t, f) \Rightarrow X(\delta^o, t, f)] \wedge B)]$, where B constitutes the occlusion-free parts of the laws. Thus, if $\Lambda(\delta, t, f) \equiv X(\delta^o, t, f)$ then δ^o is minimal; there can be no o' that satisfies the condition in (12(c)) as any such o' that is smaller than δ^o would fail to satisfy $\Lambda(\delta, t, f) \Rightarrow X(o', t, f)$.

Example: The action laws in (36) yield the following definition of X.

$$\begin{aligned}
&\forall f, t[X(\delta^o, t, f) \equiv \exists s, s', a[\\
&\quad V(\delta^v, s) < t \leq V(\delta^v, s')\wedge\\
&\quad D(\delta^d, V(\delta^v, s), V(\delta^v, s'), a)\wedge\\
&\quad ((a = \text{LOAD} \wedge f \in \{\text{LOADED}\})\vee\\
&\quad\ (a = \text{FIRE} \wedge H(\delta^h, V(\delta^v, s), \text{LOADED})\wedge\\
&\quad\ \ f \in \{\text{LOADED}, \text{ALIVE}\}))]]
\end{aligned} \tag{38}$$

3.2.5 Consequence relation

Finally, the consequence relation $\models_{NL}$ between an NL theory Γ_L (Law statements) and a logical combination of narrative consequence (Nec and Pos) and verification (Within) statements α is as follows. Γ_B denotes the base theory.

$$\Gamma_L \models_{NL} \alpha \text{ iff } \Gamma_B \wedge \forall\delta, m[\Gamma_L]^\delta_m \models \forall\delta[\alpha]^\delta_\epsilon \tag{39}$$

Example: If Γ_L is taken to be the laws (2–3) and α is taken to be the narrative consequence statement (4), then a consequence proof can be made as follows, using natural deduction. Start with Γ_B and the translation of Γ_L, that is to say (36), as premises. Next, show the translation of (4), that is to say (33), which is a universally quantified conditional, by assuming $Eq(d_1, d_2, \epsilon) \wedge Wb(d_2) \wedge De(@0, \text{S}4, \boldsymbol{n}, d_2)$ for arbitrary developments d_1, d_2. Now, from $De(@0, \text{S}4, \boldsymbol{n}, d_2)$ one can infer the right-hand-side of (23) instantiated with d_2, which states that the only action occurrences in the given time segment are $D(d_2^d, V(d_2^v, \text{S}1), V(d_2^v, \text{S}2), \text{LOAD})$ and $D(d_2^d, V(d_2^v, \text{S}3), V(d_2^v, \text{S}4), \text{FIRE})$, which occur in that order. From the above, $Wb(d_2)$, the definition of Wb (12) and the law statements (36) can then be inferred $H(d_2^h, V(d_2^v, \text{S}2), \text{LOADED})$ as an effect of $D(d_2^d, V(d_2^v, \text{S}1), V(d_2^v, \text{S}2), \text{LOAD})$. Further, with the aid of (38), one can infer $\forall t[V(d_2^v, \text{S}2) < t \leq V(d_2^v, \text{S}3) \Rightarrow \neg X(d_2^o, t, \text{LOADED})]$ and, with the aid of (12(d)), $H(d_2^h, V(d_2^v, \text{S}3), \text{LOADED})$. From the latter one can in addition infer $\neg H(d_2^h, V(d_2^v, \text{S}4), \text{ALIVE})$ as an effect of $D(d_2^d, V(d_2^v, \text{S}3), V(d_2^v, \text{S}4), \text{FIRE})$. As we have now shown the conditional for arbitrary d_1, d_2, we can infer the universal (33).

Notice that the proof did not involve any reference to the second-order axioms (8)–(11). They do come in, however, in proofs for non-consequence, where they can be used to construct developments that serve as counter-examples.

4 PROPERTIES AND APPLICATIONS

Besides supporting hypothetical reasoning, the fact that narratives are objects in NL permit us to state properties of, and relations between, narratives, and in addition defining different kinds of operations.

Theorem 2 *Let Γ_L be an NL theory, and let n_1, n_2, n_3 be narrative variables (universally quantified). Then the conditions* Within(s, s', n_1), Within(s', s'', n_2) *and* Within(s'', s''', n_3) *imply the following equivalences.*

$$\begin{aligned}&\text{Nec}(\ s, s', n_1, \ \text{Nec}(\ s', s'', n_2, \ \alpha\ )\ ) \equiv\\ &\quad \text{Nec}(\ s, s'', \text{J}[n_1, n_2, \text{T}], \ \alpha\ )\end{aligned} \tag{40}$$

$$\begin{aligned}&\text{Pos}(\ s, s', n_1, \ \text{Pos}(\ s', s'', n_2, \ \alpha\ )\ ) \equiv\\ &\quad \text{Pos}(\ s, s'', \ \text{J}[n_1, n_2, \ \text{T}], \ \alpha\ )\end{aligned} \tag{41}$$

$$\begin{aligned}&\text{Nec}(\ s, s', n_1, \ \text{Nec}(\ s'', s''', n_3, \ \alpha\ )\ ) \equiv\\ &\quad \text{Nec}(\ s'', s''', n_3, \ \text{Nec}(\ s, s', n_1, \ \alpha\ )\ )\end{aligned} \tag{42}$$

$$\begin{aligned}&\text{Pos}(\ s, s', n_1, \ \text{Pos}(\ s'', s''', n_3, \ \alpha\ )\ ) \equiv\\ &\quad \text{Pos}(\ s'', s''', n_3, \ \text{Pos}(\ s, s', n_1, \ \alpha\ )\ )\end{aligned} \tag{43}$$

$$\begin{aligned}&\text{Nec}(\ s, s', n_1, \ \alpha\ ) \wedge \text{Nec}(\ s, s', n_1, \ \alpha'\ ) \equiv\\ &\quad \text{Nec}(\ s, s', n_1, \ \alpha \wedge \alpha'\ )\end{aligned} \tag{44}$$

$$\begin{aligned}&\text{Pos}(\ s, s', n_1, \ \alpha\ ) \vee \text{Pos}(\ s, s', n_1, \ \alpha'\ ) \equiv\\ &\quad \text{Pos}(\ s, s', n_1, \ \alpha \vee \alpha'\ )\end{aligned} \tag{45}$$

Proofs. Each of the equivalences can be proved by translating them to PC, and proving them using Γ_B, the translation of Γ_L and the translations of the Within-conditions as premises, and δ and m as the current development and narrative terms. α^T_δ denotes the translation of α.

Equivalence (40) is translated to
$\forall\delta'[(Eq(\delta, \delta', m) \wedge Wb(\delta') \wedge De(s, s', n_1, \delta')) \Rightarrow$
$\forall\delta''[(Eq(\delta', \delta'', \text{E}(m, s, s', n_1)) \wedge Wb(\delta'')\wedge$
$De(s', s'', n_2, \delta'')) \Rightarrow \alpha^T_{\delta''}]] \equiv$
$\forall\delta'[(Eq(\delta, \delta', m) \wedge Wb(\delta') \wedge De(s, s'', \text{J}[n_1, n_2, \text{T}], \delta')) \Rightarrow$
$\alpha^T_{\delta'}]$.
From left to right: 1. Assume the left-hand-side of the equivalence. 1.1. Assume that $(Eq(\delta, d_1, m) \wedge Wb(d_1) \wedge De(s, s'', \text{J}[n_1, n_2, \text{T}], d_1))$ holds for some arbitrary d_1. From that assumption and the def. of De and the Within-conditions, it follows that $De(s, s', n_1, d_1)$ and $De(s', s'', n_2, d_1)$. From the def. of Eq, it follows that

$Eq(d_1, d_1, \text{E}(m, s, s', n_1))$. Now, both the outer and inner antecedents of assumption 1 are true for d_1, so we can infer $\alpha^T_{d_1}$. Exit subproof 1.1 and generalize to $\forall\delta'[(Eq(\delta,\delta',m) \wedge Wb(\delta') \wedge De(s,s',\text{J}[n_1,n_2,\text{T}],\delta')) \Rightarrow \alpha^T_{\delta'}]$. End of subproof 1.
From right to left: 2. Assume the right-hand-side of the equivalence. 2.1. Assume $(Eq(\delta,d_1,m) \wedge Wb(d_1) \wedge De(s,s',n_1,d_1))$.
2.1.1. Assume $(Eq(d_1,d_2,\text{E}(m,s,s',n_1)) \wedge Wb(d_2) \wedge De(s',s'',n_2,d_2))$.
From $Eq(\delta,d_1,m)$, $Eq(d_1,d_2,\text{E}(m,s,s',n_1))$ and the def. of Eq it follows that $Eq(\delta,d_2,m)$. $Wb(d_2)$ is already given. From $De(s,s',n_1,d_1)$, the def. of Eq and $Eq(d_1,d_2,\text{E}(m,s,s',n_1))$ it follows that $De(s,s',n_1,d_2)$, and from that, $De(s',s'',n_2,d_2)$ and the def. of De it follows that $De(s,s'',\text{J}[n_1,n_2,\text{T}],d_2)$. Now, the antecedent of assumption 2 is true for d_2, so we can infer $\alpha^T_{d_2}$. Exit subproof 2.1.1 and 2.1, and generalize, and we have $\forall\delta'[(Eq(\delta,\delta',m) \wedge Wb(\delta') \wedge De(s,s',n_1,\delta')) \Rightarrow \forall\delta''[(Eq(\delta',\delta'',\text{E}(m,s,s',n_1)) \wedge Wb(\delta'') \wedge De(s',s'',n_2,\delta'')) \Rightarrow \alpha^T_{\delta''}]]$. End of subproof 2. Consequently, the equivalence holds.

Equivalence (42) is translated to
$\forall\delta'[(Eq(\delta,\delta',m) \wedge Wb(\delta') \wedge De(s,s',n_1,\delta')) \Rightarrow$
$\forall\delta''[(Eq(\delta',\delta'',\text{E}(m,s,s',n_1)) \wedge Wb(\delta'') \wedge$
$De(s'',s''',n_3,\delta'')) \Rightarrow \alpha^T_{\delta''}]] \equiv$
$\forall\delta'[(Eq(\delta,\delta',m) \wedge Wb(\delta') \wedge De(s'',s''',n_3,\delta')) \Rightarrow$
$\forall\delta''[(Eq(\delta',\delta'',\text{E}(m,s,s',n_3)) \wedge Wb(\delta'') \wedge$
$De(s,s'',n_1,\delta'')) \Rightarrow \alpha^T_{\delta''}]]$.
From left to right: 1. Assume the left-hand-side of the equivalence. 1.1. Assume $(Eq(\delta,d_1,m) \wedge Wb(d_1) \wedge De(s'',s''',n_3,d_1))$.
1.1.1. Assume $(Eq(d_1,d_2,\text{E}(m,s'',s''',n_3)) \wedge Wb(d_2) \wedge De(s,s',n_1,d_2))$. From the definition of Eq, 1.1. and 1.1.1. it follows that $Eq(\delta,d_2,m)$. $Wb(d_2)$ and $De(s,s',n_1,d_2)$ are already in 1.1.1. $Eq(d_2,d_2,\text{E}(m,s,s',n_1))$ follows from the definition of Eq. From $Eq(d_1,d_2,\text{E}(m,s'',s''',n_3))$ and $De(s'',s''',n_3,d_1)$ it follows that $De(s'',s''',n_3,d_2)$. Thus, both the outer and the inner antecedents in assumption 1 are true for d_2. Therefore, we can derive $\alpha^T_{d_2}$. Exit subproofs 1.1.1. and 1.1. and generalize, and we have $\forall\delta'[(Eq(\delta,\delta',m) \wedge Wb(\delta') \wedge De(s'',s''',n_3,\delta')) \Rightarrow \forall\delta''[(Eq(\delta',\delta'',\text{E}(m,s,s',n_3)) \wedge Wb(\delta'') \wedge De(s,s'',n_1,\delta'')) \Rightarrow \alpha^T_{\delta''}]]$. End of subproof 1. From right to left is exactly the same; just switch the narrative and temporal terms.
Equivalence (44) is translated to $(\forall\delta'[(Eq(\delta,\delta',m) \wedge Wb(\delta') \wedge De(s,s',n_1,\delta')) \Rightarrow \alpha^T_{\delta'}]] \wedge (\forall\delta'[(Eq(\delta,\delta',m) \wedge Wb(\delta') \wedge De(s,s',n_1,\delta')) \Rightarrow (\alpha')^T_{\delta'}]]) \equiv$
$\forall\delta'[(Eq(\delta,\delta',m) \wedge Wb(\delta') \wedge De(s,s',n_1,\delta')) \Rightarrow (\alpha^T_{\delta'} \wedge (\alpha')^T_{\delta'})]]$ and should be obvious.
The remaining equivalences (41), (43), (45) follow immediately from (40), (42), (44) respectively and the definition of $\text{Pos}(s,s',n,\alpha)$ as $\neg\text{Nec}(s,s',n,\neg\alpha)$.

An important potential application of a formalism that supports reasoning about alternative choices of actions is to formalize higher-level reasoning tasks such as explanation and planning. Both the concept of an explanation and the concept of a plan can be readily formalized in NL, as illustrated by the following definitions. Explanation is the task, given some observation α at an initial time s_1 and some observation β at a later time s_2, to find a narrative n that explains how β came about.

$$\begin{aligned}&\text{Expl}(\alpha,\beta,s_1,s_2,n) \overset{\text{def}}{=} \\ &\quad \text{Pos}(s_1,s_2,n,\alpha \wedge \beta) \wedge \text{Within}(s_1,s_2,n)\end{aligned} \tag{46}$$

For instance, the death of the turkey can be explained as follows.

$$\begin{aligned}&\text{Expl}(\text{H}(@0,\text{ALIVE}), \neg\text{H}(\text{S3},\text{ALIVE}), @0, \text{S3}, \\ &\quad \text{J}[\text{D}(\text{S1},\text{S2},\text{FIRE}),(@0<\text{S1})\wedge(\text{S2}\leq\text{S3})])\end{aligned} \tag{47}$$

However, a resurrection of the turkey would be inexplicable.

$$\begin{aligned}&\neg\exists n[\text{Expl}(\neg\text{H}(\text{S0},\text{ALIVE}), \text{H}(\text{S3},\text{ALIVE}), \\ &\quad \text{S0}, \text{S3}, n)]\end{aligned} \tag{48}$$

Similarly, plan synthesis is the task, given some observation α at an initial time s_1 and some goal β at a later time s_2, to find a narrative n that is guaranteed to achieve the goal.

$$\begin{aligned}&\text{Plan}(\alpha,\beta,s_1,s_2,n) \overset{\text{def}}{=} \\ &\quad \text{Nec}(s_1,s_2,n,\alpha \Rightarrow \beta) \wedge \text{Pos}(s_1,s_2,n,\text{T}) \wedge \\ &\quad \text{Within}(s_1,s_2,n)\end{aligned} \tag{49}$$

For instance, the following is a plan for killing the turkey that will work independently of the initial state.

$$\begin{aligned}&\text{Plan}(\text{T}, \neg\text{H}(\text{S4}, \text{ALIVE}), \text{S0}, \text{S4}, \\ &\quad \text{J}[\text{D}(\text{S1},\text{S2},\text{LOAD}),\text{D}(\text{S3},\text{S4},\text{FIRE}), \\ &\quad (\text{S0}<\text{S1})\wedge(\text{S2}\leq\text{S3})])\end{aligned} \tag{50}$$

More elaborate definitions involving, for instance, preferences between explanations or plans and constraints on the contents of explanations and plans are also possible. Further, the fact that plans are composite objects in NL facilitates the description of operations such as merging or modifying plans.

It has been argued [23] that deduction-based plan synthesis, i.e. plan synthesis as a deductive existence proof (like in SC), does not require any consistency checks. It should therefore be preferable to abduction-based plan synthesis, i.e. plan synthesis as finding a sentence that entails the goal (like in most narrative-based approaches) and that has to be checked for consistency. Yet, notice that the definition of a plan above involves a narrative consistency condition Pos(s_1, s_2, n, T). Narratives in NL are richer objects than action sequences in SC and can involve ordering constraints, and it is possible to construct narratives (e.g. J[(s1<s2)∧(s2<s1)]) that do not describe any possible development of the world. The same relation holds also for abduction-based approaches (and the operator-based approaches that are common in the planning literature): totally ordered plans (sequences as in SC) are consistent by virtue of their form (provided the actions involved are consistent) whereas partially order plans can contain non-satisfiable ordering constraints and therefore have to be checked for (logical) consistency. Therefore, it is doubtful whether anything is gained in terms of efficiency by formulating plan synthesis deductively instead of abductively.

Nevertheless, a formalism like NL can be of great value for representing plan synthesis and other reasoning tasks. Further, it can be valuable as a compact representation of procedural information, that is information of what actions to execute to solve specific tasks. For instance, (50) provides a procedure for killing turkeys. With such knowledge explicitly represented, an agent does not have to construct a plan from scratch each time it wants to kill a turkey, and in particular it does not have to infer details of the plan such as what holds at what time-point. Further, by extending NL narratives with operations for, for instance, tests and conditional choices, NL could serve as a language for formalizing task execution languages such as Firby's RAPS [8], and even for reasoning about tasks (e.g. planning) in such a language. How to do this is an interesting direction for future work.

5 CONCLUSIONS

In order to provide a suitable foundation for theories about such things as plans and explanations in dynamic environments, there are a number of features that are desirable in a formalism for action and change. First, there is often the need for metric time, and for expressing nontrivial temporal properties and relations. Second, it is desirable to have support for reasoning about alternative choices of actions, and to consider such choices as objects in the logic. Third, the alternative ways the world can develop relative to a specific choice of actions should also be represented on the object level. As we stated in the introduction, these three features have been integrated in NL, and it is the introduction of narratives as objects that makes this integration possible. Narratives are entities that are separate from the underlying metric time structure, and this makes it possible to encode rich temporal relations. Due to the narrative-time separation, it should in principle also be possible to change the underlying temporal structure (e.g. to intervals) without any major modifications. Further, the fact that narratives are objects in NL makes it possible to actually reason *about* narratives as opposed to reasoning *within* a specific narrative which is the case in most other narrative-based formalisms. The encoding of the Nec and Pos operators in terms of quantification over developments makes it possible to reason about necessary and possible consequences of a narrative.

It has previously been shown that event-based branching time is not always sufficient for reasoning about alternative choices of actions [10]. In this paper, we have shown that it is not necessary for that purpose either. Actually, we claim that the temporal structure is of secondary importance for the ability to reason about alternative action choices.

NL is defined as a macro language, which is then mapped down to a PC theory and complemented with a number of PC axioms (some of them second-order). Its basis in standard logic is one of many things that distinguishes NL from the work of Pelavin [19], which is the only work with somewhat similar intentions that we are aware of. Other related work, besides what is mentioned in the introduction, includes approaches to combining linear and branching time, such as work by McDermott [17], Sandewall [24], and efforts to bridge the gap between existing narrative-based and branching event-based formalisms [18, 13, 4]. Finally, the solution to the frame problem in NL is an implementation of the PMON entailment policy [24, 6]. PMON has formally been shown to be applicable to worlds with non-overlapping actions with duration and context-dependent and nondeterministic effects, and has recently been extended to deal with ramification [9], qualification [7] and concurrency [11].

Acknowledgments

This work has been supported by grants from the Wallenberg Foundation and the Swedish Council for Engineering Sciences (TFR). Many thanks to Andrzej Szalas, Patrick Doherty, Silvia Coradeschi, Patrik Haslum and the anonymous referees for comments and advice.

References

[1] Allen, Kautz, Pelavin, and Tenenberg, editors. *Reasoning About Plans.* Morgan Kaufmann, 1991.

[2] James F. Allen. Temporal reasoning and planning. In Allen et al. [1].

[3] Andrew B. Baker. A simple solution to the Yale shooting problem. In *Principles of Knowledge Representation and Reasoning: Proceedings of the First International Conference.* Morgan Kaufmann, 1989.

[4] Kristof van Belleghem, Marc Denecker, and Danny de Schrege. On the relation between situation calculus and event calculus. *Journal of Logic and Computation, Special issue: reasoning about action and change*, 31(31: 1–3), April–June 1997.

[5] Ernest Davis. Knowledge preconditions for plans. *Journal of Logic and Computation*, 4(5):721–766, October 1994.

[6] Patrick Doherty. Reasoning about action and change using occlusion. In *Proceedings of the Eleventh European Conference on Artificial Intelligence.* John Wiley & Sons, 1994.

[7] Patrick Doherty and Jonas Kvarnström. Tackling the qualification problem using fluent dependency constraints: Preliminary report. In *TIME'98*, Florida, 1998.

[8] James Firby. *Adaptive Execution in Complex Dynamic Worlds.* PhD thesis, Yale University, 1989.

[9] Joakim Gustafsson and Patrick Doherty. Embracing occlusion in specifying the indirect effects of actions. In KR96 [14].

[10] Lars Karlsson. Reasoning with incomplete initial information and nondeterminism in situation calculus. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, August, 1997. AAAI press.

[11] Lars Karlsson and Joakim Gustafsson. Reasoning about action in a multi-agent environment. Submitted for Journal Publication. Available at Linköping University Electronic Press, `http://www.ep.liu.se/`, 1997.

[12] R. A. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 4:67–95, 1986.

[13] Robert Kowalski and Fariba Sadra. Reconciliating the event calculus with the situation calculus. *Journal of Logic and Computation, Special issue: reasoning about action and change*, 31(31: 1–3), April–June 1997.

[14] *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference.* Morgan Kaufmann, 1996.

[15] Hector J. Levesque. What is planning in the presence of sensing? In *Proceedings of the Thirteenth National Conference on Artificial Intelligence.* AAAI Press, Menlo Park, California, 1996.

[16] John McCarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502, 1969.

[17] Drew McDermott. Reasoning about plans. In J.R. Hobbs and R.C. Moore, editors, *Formal Theories of the Commonsense World.* Ablex, Norwood, New Jersey, 1985.

[18] Rob Miller and Murray Shanahan. Narratives in situation calculus. *Journal of Logic and Computation*, 4(5), October 1994.

[19] Richard N. Pelavin. Planning with simultaneous actions and external events. In Allen et al. [1].

[20] Javier A. Pinto. *Temporal Reasoning in the Situation Calculus.* PhD thesis, University of Toronto, 1994.

[21] V. Pratt. Semantical considerations on Floyd-Hoare logic. In *Proceedings of the 17th IEEE Symposium on Foundations of Computer Science*, 1976.

[22] Ray Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 359–360. Academic Press, 1991.

[23] Ray Reiter. Natural actions, concurrency and continuous time in the situation calculus. In KR96 [14].

[24] Erik Sandewall. *Features and Fluents.* Oxford Press, 1994.

Combining Narratives

John McCarthy
Computer Science Department
Stanford University
Stanford, CA 94305

Tom Costello
Computer Science Department
Stanford University
Stanford, CA 94305

Abstract

A theory is *elaboration tolerant* to the extent that new information can be incorporated with only simple changes. The simplest change is conjoining new information, and only conjunctive changes are considered in this paper. In general adding information to a theory should often change, rather than just enlarge, its consequences, and this requires that some of the reasoning be non-monotonic.

Our theories are narratives—accounts of sets of events, not necessarily given as sequences. A narrative is elaboration tolerant to the extent that new events, or more detail about existing events, can be added by just adding more sentences.

We propose a new version of the situation calculus which allows information to be added easily. In particular, events concurrent with already described events can be introduced without modifying the existing descriptions, and more detail of events can be added. A major benefit is that if two narratives do not interact, then they can be consistently conjoined.

1 OBJECTIVES OF SITUATION CALCULUS

The logical approach to AI ([McC59] and [McC89]) is to make a computer program that represents what it knows about the world in general, what it knows about the situation it is in, and also its goals, all as sentences in some mathematical logical language. The program then infers logically what action is appropriate for achieving its goal and does it. Since 1980 it has been widely known that non-monotonic inference must be included. The actions our program can perform include some that generate sentences by other means than logical inference, e.g. by observation of the world or by the use of special purpose non-logical problem solvers.

Simpler behaviors, e.g. actions controlled by servomechanisms or reflexes can be integrated with logic. The actions decided on by logic can include adjusting the parameters of ongoing reflexive actions. Thus a person can decide to walk faster when he reasons that otherwise he will be late, but this does not require that reason control each step of the walking.[1]

Situation calculus is an aspect of the logic approach to AI. A situation is a snapshot of the world at some instant. Situations are *rich*[2] objects in that it is not possible to completely describe a situation, only to say some things about it. From facts about situations and general laws about the effects of actions and other events, it is possible to infer something about future situations. Situation calculus was first discussed in [McC63], but [MH69] was the first widely read paper about it.

In this formalization of action in situation calculus, there are at least three kinds of problem—*narrative*, *planning* and *prediction*. Of these, narrative seems to be the simplest for humans. A narrative is an account of what happened. We treat it by giving some situations and some events and some facts about them and their relations. Situations in a narrative are partially ordered in time. The real situations are totally ordered[3], but the narrative may not include full information about this ordering. Thus the temporal relations between situations need only be described to the extent needed to describe their interactions. Situations occurring entirely in different places give the most obvious examples, but even actions by the same person in the same place may not interact as far as the inferences we draw. If

[1]Thus we protect our flank from the disciples of Rod Brooks.

[2]Though rich, situations are still *approximate, partial objects*. The idea will be developed elsewhere.

[3]Hypothetical situations need not be totally ordered; the situation where Oswald missed Kennedy is neither in the past nor the future.

we state that the traveler on certain flight reads a book and also drinks a Coca-cola, we humans don't need to know any temporal relations between the two events unless they interact.

In situation calculus as it was originally envisaged (and has been used,) events (mainly actions) in a situation produce next situations, e.g. $s' = Result(e, s)$. The original theory did not envisage more than one event occurring in a situation, and it did not envisage intermediate situations in which events occur. However, rarely did people write axioms that forbade[4] these possibilities; it's just that no-one took advantage of them.

Our present formalism doesn't really change the basic formalism of the situation calculus much; it just takes advantage of the fact that the original formalism allows treating concurrent events even though concurrent events were not originally supposed to be treatable in that formalism. Gelfond, Lifschitz and Rabinov [GLR91] treat concurrent events in a different way from what we propose here.

In a narrative, it is not necessary that what is said to hold in a situation be a logical consequence (even non-monotonically) of what was said to hold about a previous situation and known common sense facts about the effects of events. In the first place, in stories new facts about situations are often added, e.g. "When Benjamin Franklin arrived in London it was raining". In the second place, we can have an event like tossing a coin in which neither outcome has even a non-monotonic preference.[5]

In interpreting the following formalizations, we regard situations as rich objects and events as poor. In fact, we are inclined to take a deterministic view within any single narrative. In principle, every event that occurs in a situation and every fact about following situations is an inevitable consequence of the facts about the situation. Thus it is a fact about a situation that a coin is tossed and that it comes up tails. However, such facts are only occasionally consequences of the facts about the situation that we are aware of in the narrative.

Perhaps narrative seems easy, since it is not yet clear what facts must be included in a narrative and what assertions should be inferable from a narrative. We have however a basic model that handles some of the more basic features.

[4]Reiter [Rei93] did write such axioms.

[5]Nevertheless, some narratives are anomalous. If we record that Junior flew to Moscow, and, in the next situation mentioned, assert that he is in Peking, a reader will feel that something has been left out.

We want to introduce a concept of a *proper narrative*, this is a narrative without anomalies. The fluents holding in a new situation should be reasonable outcomes of the events that have been reported, except for those fluents which are newly asserted, e.g. that it was raining in London when Franklin arrived.

2 ELABORATION TOLERANT REASONING

A formalization of a phenomenon is *elaboration tolerant* to the extent that it permits elaborations of the description without requiring completely redoing the basis of the formalization. In particular, it would be unfortunate to have to change the predicate symbols. Ideally the elaboration is achieved by adding sentences, rather than by changing sentences. Often when we add sentences we need to use some form of non-monotonic reasoning. This is because we often want to add information that we would previously have assumed was false. Unless we use non-monotonicity we would get inconsistency. In this paper we concentrate on the easier case when there is no need for non-monotonicity.

Natural language descriptions of phenomena seem to be more elaboration tolerant than any existing formalizations. Here are the two major kinds of elaboration tolerance that we examine in this paper.

2.1 NON-INTERACTING EVENTS

Allowing the addition of a description of a second phenomenon that doesn't interact with the first. In this case the conclusions that can be drawn about the combined narrative are just the conjunction of the conclusions about the component narratives. To infer the obvious consequences of events we need to assume that some other events do not occur. In this paper, a major novelty is that we do not assume that no other events occur. We only state that there are no events that would cause an event in our narrative to fail. Thus a narrative about stacking blocks will state that the only block moving actions[6] are those mentioned. A block stacking narrative will not say that no traveling events occur. Nor will a narrative about traveling makes claims about what block stacking events happen. This allows non-interacting narratives to be consistently conjoined.

Previous proposals could not conjoin two narratives, as they either assumed that the events that happened were picked out by the result function, or they assumed that the only events that occurred were those mentioned.

2.1.1 DETAIL OF EVENTS

We can add details of an event. On the airplane from Glasgow to London, Junior read a book and drank a Coca-Cola. If we make the assumption that other relevant events do

[6]More precisely, no other actions that would move the blocks mentioned in the narrative occur. Other blocks might be stacked in Baghdad, if our narrative is about New York. Perhaps a theory of context, that would interpret a statement about all blocks in our narrative, as a statement about all the blocks *in New York* could be used here.

not happen, we can elaborate by adding another event, so long as it is compatible with what we have said. However the notion of relevant must be formalized very carefully, as is apparent when we elaborate a particular event as a sequence of smaller events. "How did he buy the Kleenex? He took it off the shelf, put it on the counter, paid the clerk and took it home." A narrative that just mentions buying the Kleenex should not exclude this particular elaboration. Moreover, if we elaborate in this way, we don't want to exclude subsequent elaboration of component events, e.g. elaborating paying the clerk into offering a bill, taking the change, etc. Our formalism allows details of an event to be added by conjoining extra sentences.

3 MODIFYING THE SITUATION CALCULUS

Formalisms such as the situation calculus of McCarthy and Hayes [MH69], and the event calculus of Kowalski [KS97] have been used to represent and reason about a changing world c.f [Sha97]. Neither of these formalisms is exactly what is needed to represent the kind of narratives we wish to consider.

The situation calculus in its most limited version does not allow us to represent what events occur explicitly—rather every sequence of events is assumed to occur. We can specify that a particular sequence of events occurs by introducing a predicate, *actual* true of just the sequence of situations that occur[7]. This is not ideal, as it forces us to decide what events happened earlier, before we name the events that happen later.

For this reason we use a modified situation calculus, adding a new predicate $Occurs(e, s)$, that states what events occur. Thus, rather than the function $Result(e, s)$ serving two purposes, stating that e occurred at s, and designating the resulting situation, we split these two functions. We keep $Result(e, s)$, but it now only denotes the result of doing e in s when e *Occurs* at s. If e does not occur, then the value of this function is an arbitrary situation[8]. This adds an event calculus style of presentation to the underlying situation calculus formalism. In particular, it allows us to specify a sequence of events, without making any claims as to what other events may have happened in the meantime.

3.1 OUR ONTOLOGY OF SITUATIONS

Reiter has suggested that the situations in the situation calculus be defined axiomatically. He suggests the following four axioms[9]

$$\begin{array}{l}
\mathbf{S0}: \forall s. \neg s < S_0 \\
\forall a, s, s'. s < Result(a, s') \equiv (s = s' \lor s < s') \\
\mathbf{P}: \forall a, a', s, s'. Result(a, s) = Result(a', s') \rightarrow \\
\quad a = a' \land s = s' \\
\mathbf{Ind}: \forall \phi. (\phi(S_0) \land (\forall a. \phi(s) \rightarrow \phi(Result(a, s)))) \rightarrow \\
\quad \forall s. \phi(s).
\end{array}$$

which determine equality of situations, relative to equality of events or actions. These axioms are categorical, that is relative to an interpretation of equality of actions, there is a unique model of situations.

Rather than use these axioms, which state that no other situations exist between s and $Result(a, s)$, we choose to say that situations can be ordered by a $<$ predicate, which is a strict partial order, which we axiomatize as follows.

$$\begin{array}{l}
\forall a, s. s < Result(a, s), \\
\forall s, s', s''. s < s' \rightarrow \neg(s' < s), \\
\forall s, s, s', s''. s < s' \land s' < s'' \rightarrow s < s''
\end{array} \tag{1}$$

The predicate $<$ is similar to the $future(s, s')$ predicate, introduced by [MH69], which is true when s' is in the future of s. We find it useful to write this in infix notation, and to use $s \leq s'$ as the non-strict version. It also is useful to write $s \leq s' \leq s''$ for $s \leq s' \land s' \leq s''$.

4 SPECIFYING THE EFFECTS OF EVENTS

In the situation calculus it is usual to specify the effects of actions by writing *effect* axioms, like[10],

$$\forall s. Holds(Loaded, s) \rightarrow Holds(Dead, Result(Shoot, s)).$$

If we move to a formalism that allows other events to occur between s and $Result(e, s)$, then this way of specifying change needs to be adjusted. It is possible that something might occur in the time between s and $Result(Shoot, s)$ that causes the event to have a different result. For this reason it seems natural to allow the preconditions, those things that hold on the left hand side, to mention properties of all times between s and $Result(Shoot, s)$.

In previous versions of the situation calculus the preconditions for an event were always modeled as a set of fluents, namely those fluents that had to hold at s, for the event to have an effect at $Result(a, s)$. If we allow other things to

[7]Pinto and Reiter [PR95] actually do this.

[8]We could choose instead to make *Result* a partial function, but this introduces the difficulties of partial functions.

[9]Reiter's notation differ from ours, he uses $do(a, s)$, while we use $Result(a, s)$. We use $\leq s'$ as a shorthand for $s < s' \lor s = s'$. Reiter writes $<$ as $\sqsubset$.

[10]As is customary in Logical A.I. we write $Holds(Dead, s)$ without saying who is dead. We can suppose the events occur in a context and lifting rules exist to make this $Dead(Victim)$ in an outer context. The outer context may contain further preconditions, like that shooter is present.

happen during an event, we cannot just specify the preconditions that must hold at the beginning of the event.

Consider a plane journey from Glasgow to London. It is necessary that the plane be in working order for the entire flight. It is also necessary to be in Glasgow at the beginning of the flight, but clearly, there is no need for this precondition to persist for the entire flight. It is necessary to have a ticket, until the airline steward takes it from you. This is an example of a precondition, "having a ticket" that must hold neither just at the moment the event starts, nor for the entire duration.

Consider another example from the Yale Shooting Problem. In order to successfully shoot a person, the gun must be loaded when the trigger is pulled, but the target must remain in the cross-hairs until the bullet hits. We represent the fact the target is in the cross-hairs by *aimed*. Thus we write:

$$\begin{array}{l}\forall s.Occurs(Shoot, s) \wedge Holds(Loaded, s) \wedge \\ \quad \begin{pmatrix}\forall s''.s \leq s'' \wedge s'' < Result(Shoot, s) \rightarrow \\ \quad Holds(Aimed, s'')\end{pmatrix} \rightarrow \\ \quad Holds(Dead, Result(Shoot, s)).\end{array}$$

A possible objection to this example is that if the target arrives in the cross-hairs at any time before impact and remains in the path of the bullet, then they will be killed. In this case we write,

$$\begin{array}{l}\forall s.Occurs(Shoot, s) \wedge Holds(Loaded, s) \wedge \\ \quad \begin{pmatrix}\exists s_1.s_1 < Result(Shoot, s) \wedge \forall s''.s_l \leq s'' \wedge \\ \quad s'' < Result(Shoot, s) \rightarrow Holds(Aimed, s'')\end{pmatrix} \rightarrow \\ \quad Holds(Dead, Result(Shoot, s)).\end{array}$$

However, both these examples show the need for preconditions to be richer than a statement of what properties hold just when the event occurs. What possible properties can occur as preconditions is important because we wish to know what kinds of axioms can occur as effect axioms.

Before we consider how to represent preconditions, we recall how we represented all possible preconditions earlier. If we wish to introduce a predicate that can parameterize all effect axioms in the old-fashioned situation calculus can write[11],

$$\begin{array}{l}\forall a, f, g.Changes(a, f, g) \overset{def}{=} \forall s.Holds(g, s) \rightarrow \\ \neg(Holds(f, s) \equiv Holds(f, Result(a, s))\end{array}$$

following [Cos97]. g is a predicate on fluents that encodes the preconditions. Parameterizing all effect axioms allows us to minimize effect axioms easily.

However, as we want to have preconditions that can extend over the duration of the event, we need more than one set of fluents. For this reason we allow as preconditions, two sets of fluents. The first set need only hold at the start of an event, while the others must persist for the entire event.

It might seem that this does not model preconditions that need hold for only part of the duration of the event. However, we can model these by using other defined fluents. Thus we can write that "It is necessary to have a ticket, until the airline steward takes it from you." using a new fluent F_1 defined by

$$\begin{array}{l}\forall s.Holds(F_1, s) \equiv Holds(Has(Ticket), s) \vee \\ \quad Holds(Takenby(Steward, Ticket), s).\end{array}$$

This fluent should hold for the entire[12] duration of the flight. $HoldsD(f, s, e)$ is a shorthand for $\forall s'.s \leq s' < Result(e, s) \rightarrow Holds(f, s')$.

We can write "It is necessary that the plane be in working order for the entire flight" using the fluent $WorkingOrder$,

$$\begin{array}{l}\forall s, l, l', t.HoldsD(WorkingOrder, s, Fly(l, l', t)) \wedge \\ HoldsD(F_1, s, Fly(l, l', t))) \rightarrow \\ \quad Holds(at(l'), Result(Fly(l, l', t), s))\end{array}$$

We can add " It is necessary to be in Glasgow[13] at the beginning of the flight"

$$\begin{array}{l}\forall s, l, l', t.Holds(at(l), s) \wedge \\ HoldsD(WorkingOrder, s, Fly(l, l', t)) \wedge \\ \quad HoldsD(F_1, s, Fly(l, l', t))) \rightarrow \\ \quad\quad Holds(at(l'), Result(Fly(l, l', t), s))\end{array}$$

Now that our preconditions are represented as two sets[14], rather than one, we redefine $Changes$ as follows:

$$\begin{array}{l}\forall e, f, g_1, g_2.Changes(e, f, g_1, g_2) \overset{def}{=} \\ \forall s.(\forall f_1.g_2(f_1) \rightarrow Holds(f_1, s)) \wedge \\ \quad (\forall f_1.g_1(f_1) \rightarrow HoldsD(f_1, s, e)) \rightarrow \\ \neg(Holds(f, s) \equiv Holds(f, Result(e, s)))\end{array}$$

We find it useful to introduce a predicate $Succeeds(e, f, s)$ defined as,

$$\begin{array}{l}\exists g_1, g_2.Changes(e, f, g_1, g_2) \wedge \\ Holds(g_2, s) \wedge HoldsD(g_1, s, e).\end{array} \quad (2)$$

Frame Axioms

We usually would write a frame axiom for a fluent, say $On(A, Top(B))$, block A is on the top of block B, and an action, in this case $Shoot$ as,

$$\begin{array}{l}\forall s.Holds(On(A, Top(B)), s) \equiv \\ Holds(On(A, Top(B)), Result(Shoot, s)).\end{array}$$

[11] We will slightly abuse notation and write $Holds(g, s)$ for $(\forall f'.g(f') \rightarrow Holds(f', s))$, when g is a predicate on fluents.

[12] The entire duration is taken to be up to, but not including the endpoint. It is sometimes natural that the endpoint should not be needed as a precondition.

[13] We write the general formula with a variable l for Glasgow.

[14] Here we assume that we have a fluent function $\neg$, such that $\forall s, f.Holds(f, s) \equiv \neg Holds(\neg f, s)$. In the absence of the fluent function $\neg$, we would need four sets, two for positive fluents and two for negative fluents.

However, since *Result* no longer encodes what events occurred, we need to say something like, "if no event that could change the fluent $On(A, Top(B))$ occurred in the interval between s and s' and $On(A, Top(B))$ held at s then $On(A, Top(B))$ will hold at s'." It is notable that this needs the notion of *Changes* that we introduced above. Thus we write,

$$\begin{array}{l}\textbf{Frame Axiom:}\\ \forall s, s', f. s \le s' \wedge \\ \left(\begin{array}{l}\forall s'', a. s < Result(a, s'') \le s' \rightarrow \\ \quad \neg(Occurs(a, s'') \wedge \\ \quad Succeeds(a, f, s''))\end{array}\right) \rightarrow \\ \quad (Holds(f, s) \equiv Holds(f, s'))\end{array} \tag{3}$$

This will generate all of our frame axioms if we minimize *Changes*, varying *Succeeds* and *Holds*, and allowing the domain to vary as in [Cos98b, Cos98a]. We do not consider using non-monotonic reasoning to minimize *Changes* here, as we wish to stress other issues. Thus we explicitly axiomatize the result of the minimization, much in the same way as Reiter[Rei91] uses an explanation closure axiom and an explicit statement of what events can change what fluents.

4.1 WHAT EVENTS OCCUR?

Before we consider combining narratives, we address a problem that arises in our new formalism that was not present in the earlier versions of the situation calculus.

Even frame axioms like this are not enough to allow us to carry out the simple reasoning we could carry out in previous versions of the situation calculus. We also need to know that certain events do not occur.

Consider the following example of moving a block. We have the action that moves a block, but to move a block successfully it must be clear. For instance if someone else puts another block on top of the block we are moving to, then our action will fail.

Thus our only effect axiom is the following one, which states moving block a onto block c succeeds, if a and c are clear for the entire duration, and a is not equal to c.

$$\begin{array}{l}\forall s, a, c, e, l. \\ \left(\begin{array}{l}a \ne c \wedge e = Move(a, Top(c)) \wedge \\ l \ne top(c) \wedge HoldsD(Clear(Top(a)), s, e) \wedge \\ \quad Holds(On(a, l), s) \wedge \\ \quad HoldsD(Clear(Top(c)), s, e)\end{array}\right) \rightarrow \\ \left(\begin{array}{l}Holds(On(a, Top(c)), Result(e, s)) \wedge \\ (l \ne Table \rightarrow \neg Holds(On(a, l), Result(e, s)))\end{array}\right)\end{array}$$

For this example we need some other facts about the world, these are given in an Appendix. We are also told that the only block that is not on the table is A, which is on B, and that the action of moving A to the top of C occurred at the situation $S0$.

$$\begin{array}{l}\forall a, l. Holds(On(a, l), S0) \equiv \\ (a = A \wedge l = Top(B)) \vee (a \ne A \wedge l = Table), \\ Occurs(Move(A, Top(C)), S0).\end{array}$$

We can now write our frame axiom, which in this case is,

$$\begin{array}{l}\forall s, s', a, c. s \le s' \wedge \\ \left(\begin{array}{l}\forall s'', a', e. e = Move(a, Top(a')) \wedge \\ s < Result(e, s'') \le s' \wedge a \ne c \rightarrow \\ \quad \neg\left(\begin{array}{l}Occurs(e, s'') \wedge \\ Succeeds(e, On(a, Top(a')), s'')\end{array}\right)\end{array}\right) \\ \quad \rightarrow \left(\begin{array}{l}Holds(On(a, Top(c)), s) \equiv \\ \qquad Holds(On(a, Top(c)), s')\end{array}\right)\end{array}$$

This states that a block a is on a block c in a situation s if and only if a is on c in s', so long as there is no event e, of moving moving a to $Top(a')$, which occurs, and is successful.

Some writers like to think that if an event that might change a fluent f occurs, but fails, then the fluent f should be undetermined. We can weaken our frame axiom, so that even if the event of moving a block a' to a fails, then our block a might not be clear. We write this as,

$$\begin{array}{l}\forall s, s', a, c, a', c', b'. (s \le s' \wedge \\ \left(\begin{array}{l}\forall s'', a'. s \le Result(a, s'') \le s' \rightarrow \\ \quad \neg\left(\begin{array}{l}Occurs(Move(a, a'), s'') \wedge \exists g_1, g_2. \\ \quad Changes\left(\begin{array}{l}Move(a, a'), \\ On(a, Top(c)), g_1, g_2\end{array}\right)\end{array}\right)\end{array}\right) \\ \rightarrow \left(\begin{array}{l}Holds(On(a, Top(c)), s) \equiv \\ \qquad Holds(On(a, Top(c)), s').\end{array}\right)\end{array}$$

In general we shall prefer the stronger frame axiom.

We wish to prove that

$$Holds(On(A, Top(C)), Result(Move(A, Top(C)), S0)),$$

however, we can only prove the weaker,

$$\begin{array}{l}\forall e. e = Move(A, Top(C)) \wedge \\ HoldsD(Clear(A), s, e) \wedge HoldsD(Clear(C), s, e) \rightarrow \\ \qquad Holds(On(A, Top(C)), Result(e, S0))\end{array}$$

Thus, we need to prove that A and C remain clear during the move action in order to show that the action is successful. To show that they remain clear we need to use our frame axiom. But, all we can prove is that the A block remains clear if there is no successful move action $Move(b_1, Top(A))$ that occurs, and whose result is before $Result(Move(A, C), S0)$. As we allow situations before $S0$, we can imagine that there was a move in progress that placed a block on C just after $S0$.

Thus we explicitly state that no action that might put something on A or C occurred in the interval[15], save of course

[15] We need to state that no event whose result lies in the interval

the action of putting A on the top of C.

$$\forall s, b_1, e.e = Move(b_1, Top(A)) \rightarrow \\ \neg\begin{pmatrix} Occurs(e,s) \wedge S0 \leq Result(e,s) \\ \leq Result(Move(A, Top(C)), S0) \end{pmatrix}$$

$$\forall s, b_1, e.e = Move(b_1, Top(C)) \wedge (s \neq S0 \vee b \neq b_1) \rightarrow \\ \neg\begin{pmatrix} Occurs(e,s) \wedge \quad S0 \leq Result(e,s) \\ \leq Result(Move(A, Top(C)), S0) \end{pmatrix}$$

We can now prove that

$$Holds(On(A, Top(C)), Result(Move(A, Top(C)), S0)),$$

as the above principal allows us to prove that A and C are clear for the entire interval.

We cannot prove that C remains on the Table however, as there may be events that put C on top of other blocks. These events will not make C unclear, so they do not block the action of putting A on C. To prove that C remains on the table we would need,

$$\forall s, x.\neg\begin{pmatrix} Occurs(Move(C,x),s) \wedge \\ S0 \leq Result(Move(C,x),s) \\ \leq Result(Move(A,C),S0). \end{pmatrix}$$

It is notable that there was no need to state that no other actions occurred. It sufficed[16] to say that no other events occurred that might cause a precondition of an event in our narrative to fail. The notion that we need only state that certain events did not occur becomes very important if we wish to axiomatize domains in a way that will later allow them to be conjoined. In fact, the motivating property for developing this new axiomatization was to allow separate axiomatizations, that do not interfere with each other, to be conjoined. This is not possible in the old-style situation calculus, as we explicitly list the sequence of actions that occurs. It is also not possible if we state that the only events that occur are those mentioned, as is sometimes done in narrative reasoning.

The reasoning that we did was not significantly more difficult than the usual reasoning in the situation calculus. We needed to check a few more conditions, namely that blocks remained clear, but strategies, such as goal regression[Rei91] continue to be effective.

4.2 EVENTS WITH MULTIPLE EFFECTS

In general an event may have more than one effect. The preconditions for each effect may differ, thus preconditions may be parameterized by the effect. Furthermore, each effect may occur at a different time.

4.2.1 Extending *Result*

If there is more than one effect of an event e, we write $Result(e,s)$ for the time of the main effect, and $Result(e,f,s)$ for the time of the effect of changing f. For instance, flying from Glasgow to London has as its resulting situation the situation where you arrive in London. However, another effect of this event is to no longer have your ticket, as the air-hostess takes it from you. The situation where she takes the ticket off you is picked out by $Result(Fly(Glasgow, LHR, T_1), takenby(Steward, ticket), s)$. At this situation, the fluent $Has(ticket)$ is also made false. We use other situations like the time the airline-steward takes your ticket, rather than explicit times, as explicit times, like all numerical values are less natural—the numbers are hard to get. The statement that you no longer have your ticket after the air-hostess takes it is very intuitive, while the statement that your no longer have your ticket after n minutes, for some definition of n is not.

4.2.2 Implied events

We can deal with events having multiple effects at different times by stating that certain events trigger other events. Thus we might write,

$$\forall s, l, l'.occurs(Fly(l, l', Ticket), s) \rightarrow \\ occurs(take(Steward, Ticket), s).$$

This is an alternate way to model the notion that the airline steward takes your ticket during the flight.

We now consider narratives in two domains. One concerns stacking blocks, the other a plane journey. We show that we can axiomatize these two narratives separately, but in such a way that their conjunction is consistent.

5 GLASGOW, LONDON, MOSCOW AND NEW YORK

The object of this section is to give narratives illustrating the treatment of concurrent events in two cases. The first is when two sub-narratives do not interact, and the second

or at the endpoints, thus the use of $\leq$. Sometimes, especially when we are checking preconditions of events, we will only need to show that nothing had an effect strictly before the end, and this we will only need to show $<$. When we try to use inertia we will need to show the $\leq$.

[16]In this paper we state that the other events do not happen monotonically. These statements can be inferred non-monotonically from sentences that tell which occurrences and what fluents are explicitly *stated* to occur and hold in our narrative, and the axiomatization of *Changes*. A fluent is relevant if it is a precondition or an effect of a stated event that occurs, or if the fluent's value is stated in the narrative. This gives us a notion of what the relevant fluents and events are in terms of what fluents and events are explicitly given in the narrative. We then state that no other events occur that would change the effects of the relevant fluents. We avoid explaining this reasoning, as the machinery we currently use is quite complex.

is when they do. The first sub-narrative is ordinary block stacking (as discussed in many situation calculus papers), and we suppose the stacking to be done by a person called Daddy in New York.

In the second sub-narrative, the actor is named Junior, and he wants to fly from Glasgow to Moscow via London. The story is taken from earlier web-published but widely circulated manuscripts [McC92] discussing how circumscription could be used to treat an unexpected obstacle to a plan, and [McC95] how narratives should be represented. This story is also used by Shanahan in [Sha97] in Chapter 10 as an example to motivate a use of context.

These two sub-narratives do not interact, and thus give an example of our first goal, a treatment of non-interacting narratives that can be conjoined consistently.

Because we want to treat interacting events, we make life more complicated for Junior. If he loses his ticket, he must wire Daddy in New York for money. Daddy, who normally indulges Junior, has to interrupt his block stacking and sell a block in order to get the money to send Junior. In this part of the narrative we have an example of adding details to an event. We state the event of Junior getting money occurs, we also give a sequence of events, Daddy stacking blocks until block3 is clear, then selling block 3, receiving money and sending it to Junior. The sequence *realizes* the single event of getting money. We show that both statements are consistent with each other, and the explanation can be consistently conjoined onto the narrative that mentions only the first event.

The following uses the axiomatizations of traveling and commerce and blocks-world in the Appendix. In the text we only give those axioms that are particular to the story.

We give axiomatizations of both the narratives where Junior loses his ticket, and contacts Daddy who sends him money, that Daddy raises by selling a block (In New York, blocks are made of Gold). Naturally Daddy has to clear the block before selling it, so the narratives interact in a non-trivial way.

Narrative 1

In this narrative Junior doesn't lose his tickets, T_1 and T_2 and gets to Moscow without asking for help. Daddy stacks blocks in New York. There is no interaction, and nothing is said about the time relations between the two sub-narratives.

$$\begin{array}{l} Holds(At(J, Glasgow), S0) \\ Occurs(Fly(Glasgow, LHR, T_1), S0) \\ Dest(T_1) = LHR \wedge Source(T_1) = Glasgow \\ Holds(Has(J, T_1), S0) \\ Dest(T_2) = Moscow \wedge Source(T_2) = LHR \\ Holds(Has(J, T_2), S0) \\ Result(Fly(Glasgow, LHR, T_1), S0) < S1 \end{array} \tag{4}$$

We should be able to infer:

$$Holds(At(J, LHR), S1)$$

To infer this we need to know that,

$$\begin{array}{l} \forall s', e. S0 \leq Result(e, s') < \\ Result(Does(J, Fly(Glasgow, LHR, T_1)), S0) \rightarrow \\ \neg(Occurs(e, s') \wedge Succeeds(e, Has(J, T_1), s')) \end{array}$$

That is, no event occurs that would cause Junior to lose his ticket before he has to give it to the air-hostess[17]. We actually state the following stronger fact[18], that no events that would cause Junior to no longer have a ticket occur, save of course flying from Glasgow.

$$\begin{array}{l} \forall s', e. (t = T_1 \vee t = T_2) \wedge \\ \quad (e \neq Fly(Glasgow, LHR, T_1) \vee s' \neq S0) \wedge \\ \quad S0 \leq Result(e, s') < S1 \rightarrow \\ \neg(Occurs(e, s') \wedge Succeeds(e, Has(J, t), s')) \end{array} \tag{5}$$

When Junior is in London, inertia, and the instance of the above axiom with $t = T_2$, gets us that Junior still has the ticket to Moscow. As for the ticket to London, we would infer that he does not have it as we brought up the fact that a ticket is used up when one takes the flight the ticket is for. That is certainly a part of the knowledge of anyone who travels using tickets. Thus someone who had traveled by bus would infer it about airplane travel. Indeed it could be inferred from more general principles about commerce, e.g. that a seller doesn't want to allow the buyer to get an arbitrary number of what he has a paid for one of. However, anyone who travels has the more specific information and doesn't need to infer it from general principles about commerce. Indeed he may never have formulated any general principles about commerce.

$$\begin{array}{l} Occurs(Fly(LHR, Moscow, T_2), S1) \\ Result(Fly(LHR, Moscow, T_2), S1) < S2 \end{array} \tag{6}$$

We wish to infer,

$$Holds(At(Junior, Moscow), S2)$$

Again we need to know that no bad events occur, that is, Junior doesn't lose any tickets.

$$\begin{array}{l} \forall s', e. (e \neq Fly(LHR, Moscow, T_2) \vee s' \neq S1) \wedge \\ \quad S1 \leq Result(e, s') < S2 \rightarrow \\ \neg(Occurs(e, s') \wedge Succeeds(e, Has(J, T_2), s')) \end{array} \tag{7}$$

We call these sentences $Nar1J$, that is the sentences from 4 to 7. Now we begin Daddy's life as a block stacker.

[17]If we wished that the air-hostess took Junior's ticket at another time, we might use our three argument version of result and write,

$$\begin{array}{l} \forall s', e. S0 \leq Result(e, s') < \\ Result\begin{pmatrix} Does(J, Fly(Glasgow, LHR, T_1)), \\ Has(J, T_1), S0 \end{pmatrix} \rightarrow \\ \neg(Occurs(e, s') \wedge Succeeds(e, Has(J, T_1), s')). \end{array}$$

[18]Whether or not the stronger fact is warranted depends on whether we wish to state that no event that *might* cause Junior to lose his ticket happens, or no event that *does* cause Junior to lose his ticket happens.

We have no $\leq$ relation between the situations $S0$ and $S0'$ and know nothing of their temporal relations. If we asserted $S0 < S0' < S1$, then we could conclude that Junior still had the tickets in $S0'$. Also asserting $S0' = S0$ would do no harm to the conclusions drawn about either sub-narrative.

$$\begin{array}{l} Holds(At(D, NY), S0') \\ Holds(Has(D, A_1), S0') \\ Holds(Has(D, A_2), S0') \\ Holds(Has(D, A_3), S0') \\ Holds(On(A_3, Top\ A_1), S0') \\ \forall b.Holds(Clear(b), S0') \equiv b \neq A_1 \\ Holds(On(A_1, Table), S0') \\ Holds(On(A_2, Table), S0') \\ Occurs(Does(D, Move(A_3, Table)), S0') \\ Result(Does(D, Move(A_3, Table)), S0') < S1' \\ Occurs(Does(D, Move(A_2, Top\ A_1)), S1') \\ Result(Does(D, Move(A_2, Top\ A_1)), S1') < S2' \\ Occurs(Does(D, Move(A_3, Top\ A_2)), S2') \\ Result(Does(D, Move(A_3, Top\ A_2)), S2') < S3' \end{array} \tag{8}$$

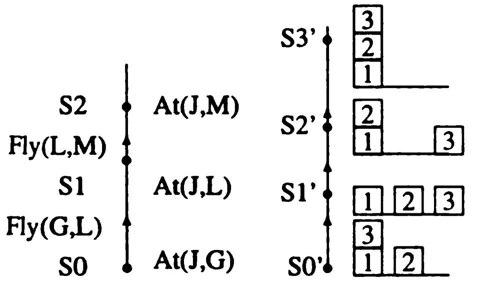
We also need to know that no other actions that would interrupt the block stacking[19] occur.

$$\begin{array}{l} \forall s', e, a, b.S0' \leq Result(e, s') \leq S3' \land \\ \quad (s' \neq S0' \lor e = Move(A_3, Table)) \land \\ \quad (s' \neq S1' \lor e = Move(A_2, Top\ A_1)) \land \\ \quad (s' \neq S2' \lor e = Move(A_3, Top\ A_2)) \rightarrow \\ \neg(Occurs(e, s') \land Succeeds(e, On(a, Top(b)), s')) \end{array} \tag{9}$$

We call the sentences from 8 to 9 $Nar1D$ We now notice that if B is the axiomatization of blocksworld in the Appendix, and T is the axiomatization of traveling, then

$$\begin{array}{l} B \land T \land Nar1D \land Nar1J \models \\ \quad Holds(On(A_3, Top\ A_2), S3') \land \\ \quad Holds(On(A_3, Table), S1') \land \\ \quad Holds(At(J, LHR), S1) \land \\ \quad Holds(At(J, Moscow), S2) \end{array} \tag{10}$$

Thus we can derive the obvious conclusions of our narrative. We further note that the two narratives are consistent.



Narrative 2

In this narrative Junior loses the ticket and sends a telegram to Daddy asking for money. Daddy, who normally indulges Junior, sells a block and sends Junior the money, who buys a London-Moscow ticket and goes on to Moscow.

We chose a telegram rather than a telephone call, because we would not want to tell about a telephone call as a sequence of statements by Junior and Daddy but rather to regard its result as a joint action, e.g. an agreement that Junior and Daddy would do certain actions.

Note also we haven't treated what Daddy now knows as the result of the telegram. It seems that treating knowledge and treating agreement are similar in their requirement for treating intentional entities. The intentional state that Junior has requested that Daddy send him the money is not merely that Daddy knows that Junior wants Daddy to send him the money. Also the agreement is likely to have something like a bit of narrative as an argument, e.g. a set of actions that Junior and Daddy will do with only partial time relations between the actions.

Here we include sentences 4 and 5. Up to here, narrative 2 is the same as narrative 1. We will also need the sentences 8 and 9.

$$Occurs(Loses(J, T_2), S1) \tag{11}$$

This contradicts 7, which stated that no event that lost the ticket happened before $S2$. We want to regard losing the ticket as something that happens to Junior rather than as something he does. That's why we don't write $does(J, lose\ ticket(LHR, Moscow))$. The bad consequences of doing the latter would arise when we get around to writing laws that quantify over voluntary actions. We will use some of the same names now for situations that are different than in narrative 1.

$$\begin{array}{l} Result(Loses(J, T_2), S1) < S2 \\ \neg Holds(Has(J, Cash), S2) \end{array} \tag{12}$$

$$Occurs(Does(J, Telegraph(D, Request\ Send\ Cash)), S2) \tag{13}$$

Here we intend to have two explanations for what happens next. One is the simple observation that Daddy does sell a block, and send the money to Junior. This is a simple sequence of events, like we detailed earlier.

However we also know that as a dutiful father, Daddy will get money to Junior. Thus we can predict the event of Daddy getting money to Junior. Here we are treating Daddy as if his actions were determined by our inputs. Sometimes it is useful to describe people in that way. In more elaborate narratives we would need to reason about Daddy mental processes, but for this case we can treat him as an automaton.

The following axiom characterizes what Daddy does when

[19]If we wish to restrict this to block stacking in New York we would add a conjunct $Holds(In(a, New\ York), s') \land Holds(In(b, New\ York), s')$ to the left hand side of the implication.

he receives a request from Junior.

$$\forall s.Holds\left(\begin{array}{l} Happens(Gets(J, Cash)), \\ Result\left(\begin{array}{l} Receives(D, Telegram\text{-}from \\ (J, Request\ Send\ Cash)), s \end{array}\right)\end{array}\right)$$

$$S3.5 = Result\left(\begin{array}{l} Receives(D, Telegram\text{-}from \\ (J, Request\ Send\ Cash)), S2 \end{array}\right) \quad (14)$$

This is an example of a triggered action, as we have the defining rule for $Holds(Happens(e), s)$,

$$\forall e, s.Holds(Happens(e), s) \equiv Occurs(e, s).$$

We now state that the money arrives before $S3$, when Junior buys the ticket.

$$Result(Gets(J, Cash), S3.5) < S3$$

We now give the other facts about occurrences.

$$\begin{array}{l} S3' = \\ Result(Does(J, Telegraph(D, Request\ Send\ Cash)), S2) \\ \neg Holds(Has(D, Cash), S3') \\ Occurs(Does(D, Sell\ A_3), S3') \\ Result(Does(D, Sell\ A_3), S3') < S4' \\ Occurs(Does(D, Send(J, Cash)), S4') \\ Result(Does(D, Send(J, Cash)), S4') < S3 \\ Occurs(Does(J, Buy\ ticket(T_2)), S3) \\ Result(Does(J, Buy\ ticket(T_2)), S3) < S4 \\ Occurs(Does(J, Fly(LHR, Moscow)), S4) \\ Result(Does(J, Fly(LHR, Moscow)), S4) < S5 \end{array}$$

We also need to know that no events occur that would divert the money in the meantimes between these events and the result of the previous events.

$$\begin{array}{l} \forall s', e.(e \neq Does(D, Send(J, Cash)) \vee s' \neq S4')\wedge \\ Result(Does(D, SellA_3), S3') \leq \\ Result(e, s') \leq S4' \rightarrow \\ \neg\left(\begin{array}{l} Occurs(e, s')\wedge \\ Succeeds(e, Has(D, Cash), s') \end{array}\right) \end{array} \quad (15)$$

$$\begin{array}{l} \forall s', e.(e \neq Does(J, Buy\ ticket(T_2)) \vee s' \neq S3)\wedge \\ Result(Does(D, Send(J, Cash)), S4') \leq \\ Result(e, s') \leq S3 \rightarrow \\ \neg\left(\begin{array}{l} Occurs(e, s')\wedge \\ Succeeds(e, Has(J, Cash), s') \end{array}\right) \end{array}$$

$$\begin{array}{l} \forall s', e.(e \neq Fly(LHR, Moscow, T_2) \vee s' \neq S4)\wedge \\ Result(Does(J, buy\ ticket(T_2)), S3) \leq \\ Result(e, s') \leq S4 \rightarrow \\ \neg\left(\begin{array}{l} Occurs(e, s')\wedge \\ Succeeds(e, Has(J, T_2), s') \end{array}\right) \end{array}$$

We now consider the consequences of narrative two. Let $Nar2$ be those sentences directly above and the sentences from 4 to 5 and 8 and 9 and 11 to 15.

$$\begin{array}{l} Nar2 \models \\ \quad \neg Holds(Has(J, T_2), S2)\wedge \\ \quad Holds(Has(D, Cash), S4')\wedge \\ \quad Holds(Has(J, Cash), S3)\wedge \\ \quad Holds(Has(J), T_2, S4)\wedge \\ \quad Holds(At(J, Moscow), S5) \end{array} \quad (16)$$

Most interestingly we can derive the occurrence of a triggered action:

$$Nar2 \wedge Dep \wedge U \models \quad (17)$$

$$Occurs(\ gets(J, Cash), S3.5\ )$$

We have two explanations for Junior receiving the money, the *gets* event, and the *send* event. We cannot tell which happens first, or if they happen simultaneously. Thus our formalism allows us to add detail of an event without contradiction.

5.1 ELABORATIONS

Interpolating unconnected situations and events into a narrative does not harm the conclusions. For example, we could put situations $S0.5$ and $S0.7$ between $S0$ and $S1$, and suppose that Junior reads a book on the airplane during the inner interval. The previous statements about what holds when Junior arrives in London should still seem ok. Indeed we have that when we add

$$\begin{array}{l} Occurs(Read(J, Book), S1.5) \\ \wedge S1 < S1.5 < Result(Read(J, Book), S1.5) < S2 \\ \forall s.Holds(Intelligent(P), Result(Read(P, Book), s)) \end{array}$$

we can conclude all our previous sentences, plus some more about Junior's intelligence, something we earlier did not have an opinion about.

In our second narrative, after $S2$ we have two possible explanations of how Junior gets the cash to buy his ticket. One explanation is that Daddy always gets cash to Junior. We also have the more detailed explanation that Daddy sells A_3 and sends the proceeds to Junior. The more detailed explanation is an elaboration of how Daddy got them cash to Junior. If is worth noting that both explanations can co-exist in our narrative without inconsistencies.

5.2 ELABORATION OF NARRATIVES

Suppose we are asked, "How did Junior fly from Glasgow to London?" and want to respond with facts about taking a taxi to the airport, presenting his ticket at the check-in counter, going to the gate, getting on the airplane, taking his assigned seat, etc. We can add this additional narrative with its intermediate situations, and we can throw in reading the book if we like. There is no reason to discard $Occurs(Does(J, Fly(Glasgow, LHR, T_1)), S0)$.

We merely have a redundant way of reaching the same conclusion. This is allowed in our formalism, and this property is demonstrated by the two ways in which we describe how Daddy gets the money for Junior.

However, we would like a sentence relating the more detailed narrative to the less detailed narrative, i.e. of asserting that one *realizes* the other. For this we will at least need narratives as objects, and this has not yet been introduced.

Note that the relation $Elaborates(N2, N1)$, when we get around to introducing it, will not be like the relation between a subroutine call and the subroutine. $N1$ will not in any sense be the definition of $N2$. $N2$ could be realized in a number of ways, only one of which corresponds to $N1$.

5.3 PLANNING AND PREDICTION

We would like to treat the circumstances of the previous narrative from the point of view of planning. In that case we need to be explicit about the consequences of actions and other events. The difference between planning and narrative is that in narrative we know that events and actions will succeed. This allows us to make assumptions we otherwise could not make. We can also assume that all the important effects of actions are mentioned. When we plan we need to show that we have taken into account all the important effects.

Let us consider the purposes of Junior and Daddy and predict what actions they will take and what the outcome will be. Of course, Junior losing the ticket will be an unpredicted event. We just throw it in, but then we should be able to predict what Junior and Daddy will subsequently do. This seems more difficult than either planning or predication.

5.4 PHILOSOPHICAL CONSIDERATIONS

Reality may be regarded as the deterministic limit of nondeterminist approximations. In what a human or robot can know about the world many events are not inevitable. In any human account, it did not have to be raining when Benjamin Franklin first arrived in London. Indeed, maybe it wasn't. Even if the world is deterministic, any achievable description of it is nondeterministic. Elaborations of particular narratives sometime remove some of the nondeterminism by accounting for the causes of particular events and for fluents holding in the results of these events.

Therefore, it may be worthwhile to regard the world as determinist and suppose that every event has causes whether we know them or not. Thus any particular nondeterminism is potentially eliminable.

It might be supposed that quantum mechanics vitiates these considerations, but we don't think it requires modifications on the common sense level. Free will in a determinist world is discussed in [MH69].

5.4.1 REMARKS

We have always felt that the careful classification of the ways in which events can overlap is unnecessary for almost all common sense reasoning. We think this article shows it. Moreover, it is also usually unnecessary to combine concurrent events into compound events as do Gelfond, Lifschitz and Rabinov [GLR91].

Acknowledgments

This research was partly funded by DARPA (ONR) grant # N00014-94-0775 under ARPI and by the Air Force Office of Scientific Research Grant # F49620-97-1-0207 under the DARPA HPKB project and New World Vistas.

References

[Cos97] Tom Costello. *Non-monotonicity and Change*. PhD thesis, Stanford University, 1997.

[Cos98a] Tom Costello. Domain Formula Circumscription. *Journal of Logic Language and Information*, 1998. to appear.

[Cos98b] Tom Costello. Minimizing the Effects of Actions. In *Proceedings of the Fifth International Symposium on Commonsense Reasoning*, 1998.

[GLR91] Michael Gelfond, Vladimir Lifschitz, and Arkady Rabinov. What are the limitations of the situation calculus? In Robert Boyer, editor, *Automated Reasoning: Essays in Honor of Woody Bledsoe*, pages 167–179. Kluwer Academic, Dordrecht, 1991.

[KS97] R. Kowalski and F. Sadri. Reconciling the Situation Calculus and Event Calculus. *Journal of Logic Programming*, 31:39–58, 1997.

[McC59] John McCarthy. Programs with Common Sense[20]. 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] J. McCarthy. Situations, Actions and Causal Laws. Technical Report Memo 2, Stanford University Artificial Intelligence Project, Stanford, CA, 1963.

[20]http://www-formal.stanford.edu/jmc/mcc59.html

[McC89] John McCarthy. Artificial intelligence, logic and formalizing common sense. In Richmond Thomason, editor, *Philosophical Logic and Artificial Intelligence*. Klüver Academic, 1989.

[McC92] John McCarthy. Overcoming an Unexpected Obstacle. Available as http://www-formal.stanford.edu/jmc/elaboration.html, 1992.

[McC95] John McCarthy. Situation Calculus with Concurrent Events and Narrative[21]. 1995. Contents subject to change. Reference will remain.

[MH69] J. McCarthy and P. Hayes. Some Philosophical Problems From the Standpoint of Artificial Intelligence. In D. Michie, editor, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, Edinburgh, UK, 1969.

[PR95] J. Pinto and R. Reiter. Reasoning about Time in the Situation Calculus. *Annals of Mathematics and Artificial Intelligence*, 14(2-4):251–268, September 1995.

[Rei91] R. Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation*, pages 359–380. Academic Press, 1991.

[Rei93] R. Reiter. Proving properties of states in the situation calculus. *Artificial Intelligence*, 64(2):337–351, December 1993.

[Sha97] Murray Shanahan. *Solving the Frame Problem, a mathematical investigation of the common sense law of inertia.* M.I.T. Press, 1997.

APPENDIX

BLOCKSWORLD

Our blocks world has 4 sorts, situations s, blocks b, locations l and actions a. These are all disjoint.

We have a situation constant $S0$, other situation constant Sn and Sn' for various n's, a set of block constants $A_1, \ldots, A_n, \ldots$, where $n \in \omega$ and one block location constant $Table$. We also have constants $A = A_1$, $B = A_2$ and $C = A_3$.

All blocks are unique, but we do not postulate domain closure.

$$A_i \neq A_j | i \neq j$$

[21] http://www-formal.stanford.edu/jmc/narrative.html

We have block locations[22], which are the Top of a block, or are the $Table$.

$$\forall l. \exists b. Top(b) = l \lor l = Table$$

All distinct block location terms denote distinct locations.

$$\forall b, b'. Top(b) = Top(b') \rightarrow b = b'$$

$$\forall b. Top(b) \neq Table$$

We have a function from actions and situations to situations, $Result(a, s)$, and a function from blocks and locations to actions, $Move(b, l)$, which gives the action where block b is moved to location l.

All distinct action terms are distinct.

$$\forall b, b', l, l'. Move(b, l) = Move(b', l') \rightarrow b = b' \land l = l'$$

We have the foundational axioms for situation calculus we considered earlier.

We have fluents, $Holds(On(b, l), s)$ which states that b is on location l in situation s, and $Holds(Clear(l), s)$. $Holds(Clear(l), s)$ is fully defined in terms of On.

$$\begin{aligned}&\forall l, s. Holds(Clear(l), s) \equiv\\ &(\exists b. \forall b'. l = Top(b') \land \neg Holds(On(b', Top(b)), s)) \lor\\ &\quad l = Table\end{aligned}$$

We now add the obvious definition of $Changes$ for $Move(b, l)$ actions. That is, there is a change in $On(b, l')$ and $On(b, l)$ when g_2 contains $On(b, l')$ for an l' not equal to l, and $l \neq Top(b)$, and g_1 contains $Clear(l)$ and $Clear(top(b))$.

$$\begin{aligned}&\forall b, b', l, l', g_1, g_2.\\ &Changes(Move(b, l), On(b', l'), g_1, g_2) \equiv\\ &l \neq l' \land l \neq Top(b) \land b = b' \land\\ &\begin{pmatrix} (b = b' \land g(\neg On(b', l'))) \lor \\ (b \neq b' \land g(On(b', l'))) \end{pmatrix} \land\\ &G(Clear(l)) \land G(Clear(l'))\end{aligned}$$

This concludes the axiomatization of blocksworld, we could add domain constraints, but this is not necessary for the reasoning we do in this paper. We now present an axiomatization of traveling, followed by an axiomatization of buying selling and sending and receiving.

TRAVELING AND COMMERCE

Our events are flying, doing actions, getting, receiving and losing. Our actions are selling, sending, telegraphing. Distinct event terms are distinct. We have two people constants, Junior and Daddy. We have among our objects Cash,

[22] These are not the same as geographical locations like New York or London. We use l to range over both, which is unfortunate.

a message ($Request\ Send\ Cash$), cities, including London, Glasgow and Moscow. We also have tickets, and functions which yield destination $Dest$ and source $Source$ of a flight, given a ticket for that flight. We have unique names for all fluent terms, and thus for all the terms that can appear in fluents. Our fluent forming functions are At which takes a person and a place, Has which takes a persona and a thing, $Happens$, which takes an event, and the earlier fluents of On and $Clear$. Our sorts are disjoint, and the sorts of variables are to be inferred by their use.

$$\begin{array}{l}\forall s, P, l, t. Holds(At(P,l),s) \wedge Source(t) = l \wedge \\ Dest(t) = l' \wedge \\ (\forall s'. s \leq s' < Result(Fly(l,l',t),s) \rightarrow \\ Holds(Has(P,t),s')) \rightarrow \\ Holds(At(P,l'), Result(Fly(l,l',t),s)) \wedge \\ \neg Holds(Has(P,t), Result(Fly(l,l',t),s))\end{array}$$

This is equivalent to

$$\begin{array}{l}\forall l, l', t, P. \\ Changes(Fly(l,l',t), Has(P,t), g_1, g_2) \leftarrow \\ \quad Source(t) = l \wedge Dest(t) = l' \wedge \\ \quad g_2(At(P,l)) \wedge g_1(Has(P,t))\end{array}$$

As this is the only effect axiom for flying, we can change this to the equivalence.

$$\begin{array}{l}\forall l, l', t, P, g_1, g_2. \\ Changes(Fly(l,l',t), Has(P,t), g_1, g_2) \equiv \\ \quad Source(t) = l \wedge Dest(t) = l' \wedge \\ \quad g_2(At(P,l)) \wedge g_1(Has(P,t))\end{array}$$

$$\forall s, P, o. \neg Holds(has(P,o), Result(lose(P,o),s))$$

This immediately gives,

$$\forall s, P, g_1, g_2, o. Changes(Lose(P,o), Has(P,o), g_1, g_2)$$

as any set of preconditions is sufficient. We add the obvious axioms that describe $Changes$ for the following effect axioms using the same method.

$$\begin{array}{l}\forall r, P, s. Holds(Happens(Receives(P', \\ \qquad\qquad Telegram\text{-}from(P,r))), \\ \qquad Result(Does(P, Telegraph(P',r)),s))\end{array}$$

$$\forall s. Holds\left(\begin{array}{l}Happens(Gets(J,Cash)), \\ Result\left(\begin{array}{l}Receives(D, Telegram\text{-}from(J, \\ \quad Request\ Send\ Cash)), s\end{array}\right)\end{array}\right)$$

$$\begin{array}{l}\forall s, P, o. Holds(Has(P,o),s) \wedge Holds(Clear(o),s) \rightarrow \\ \quad Holds(Has(P,Cash), Result(Does(P,Sell(o)),s)) \\ \forall s, P, o. Holds(Has(P,o),s) \wedge Holds(Clear(o),s) \rightarrow \\ \quad \neg Holds(Has(P,o), Result(Does(P,Sell(o)),s)) \\ \forall s, P, P', o. Holds(Has(P,o),s) \rightarrow \\ \quad Holds(Has(P',o), Result(Does(P,Send(P',o)),s)) \\ \forall s, P. Holds(Has(P,Cash), Result(Gets(P,Cash),s))\end{array}$$

How (Not) To Minimize Events

Michael Thielscher[*]
Department of Computer Science
Dresden University of Technology
01062 Dresden (Germany)

Abstract

When drawing conclusions about narratives, minimizing—to a reasonable extent—the occurrence of events is crucial. We argue that unguided minimization is insufficient in case events are causally connected, for it easily fails to distinguish unmotivated event occurrences from those that have a cause. Two solutions are offered, the first of which has the advantage of being straightforwardly realised but on the other hand has a restricted range of applicability. Our second solution overcomes these restrictions but requires two uncommon and novel features. First, event occurrences are identified as *fluents*, which allows to adapt a recent causality-oriented solution to the Ramification Problem so that if an event is caused by another event then the former is obtained as *indirect effect* of what caused the latter. Second, volitional actions and natural events which have no cause inside the reasoning context, are furnished with a special cause, namely, the reaching of the time-point at which they take place. We present both a high-level narrative description language and an axiomatisation based on a novel Fluent Calculus in which is realized this solution to the event minimisation problem.

1 INTRODUCTION

Commonsense reasoning about narratives requires, in one way or the other, to minimize the occurrence of events. For otherwise none of the intended and intuitive conclusions can be drawn. Suppose, for example, we are told that adding hot water to a mug containing a tea bag produces tea, and furthermore that a robot first places a tea bag into the empty mug and, then, pours hot water into it. The expected conclusion would be that the robot has tea afterwards. This, however, does not follow until we explicitly exclude the occurrence of a variety of events, such as the robot emptying the mug in between the two actions, someone stealing the tea bag out of the mug, or a falling tile striking the mug while water is being poured etc. Any of these events falsifies the intended conclusion, and their non-occurrence does not follow *per se* from the narrative. Hence additional measures need to be taken to conclude, by default, that none of these nor any other intervening events materialize.

As long as events are mutually independent, minimizing them wrt. a given formalised narrative in a reasonable fashion is straightforward. Once we have to consider causal dependencies among events, however, finding a good general minimisation strategy becomes a non-trivial challenge. In particular, simple global minimization is insufficient in that it may fail to produce the intended and intuitively expected conclusions. Let us illustrate this point with two simple examples, where we borrow basic notions and notations from the Event Calculus variant of [Shanahan, 1996]. The occurrence of an event e at time t is represented by the atomic expression $Happens(e,t)$, and minimisation is achieved by circumscribing predicate $Happens$ in a set of formulas which axiomatise a narrative.[1]

Example 1 Suppose a robot walking towards a wall will collide with it provided the robot does not stop beforehand. Ignoring precise values for distance and speed etc., let the formula

$$\begin{array}{l} Happens(\mathtt{walk},t) \land \neg Happens(\mathtt{stop},t+1) \\ \quad \supset Happens(\mathrm{bump},t+2) \end{array} \qquad (1)$$

encode this knowledge. Consider, now, a narrative which consists of the sole fact that the robot starts walking towards the wall at time $t=5$,

[*] On leave from Darmstadt University of Technology.

[1] It should be stressed, however, that the problems we elaborate in the following are of general nature and so do also emerge in other than this specific approach.

that is, $Happens(\mathtt{walk}, 5)$. What would one expect to happen? Since nothing indicates that the robot stops walking at time $t = 6$, the natural conclusion would be that $\neg Happens(\mathtt{stop}, 6)$, hence $Happens(\mathtt{bump}, 7)$. Minimizing *Happens* in the formula $N = Happens(\mathtt{walk}, 5) \wedge (1)$, however, does not suffice for this entailment. Rather it results in two kinds of models, one with the intended course of events and one where the opposite is true, viz. $Happens(\mathtt{stop}, 6)$ and $\neg Happens(\mathtt{bump}, 7)$. ■

Example 2 Suppose that if the left hand side of a table is lifted but not the right hand side, then soup spills out of a bowl sitting on the table; the same happens if the right hand side only is lifted:

$$\begin{aligned} Happens(\mathtt{lleft}, t) \equiv \neg Happens(\mathtt{lright}, t) \\ \supset Happens(\mathtt{spills}, t) \end{aligned} \qquad (2)$$

Suppose we are told that the left hand side of the table is lifted at time $t = 2$, i.e., $Happens(\mathtt{lleft}, 2)$. Then we should expect soup spilling out rather than a magical simultaneous lifting of the right hand side. As before, however, circumscribing *Happens* yields two indistinguishable kinds of models and so does not allow this conclusion. ■

In both our two scenarios, unguided minimization does not yield the reasonably expected conclusions because it is too weak a minimisation strategy.[2] In the following section, we show how a more elaborated minimisation strategy is obtained by formally introducing the general distinction between *actions* (which are volitional, i.e., involve a free-will decision) and *natural events*. This strategy resembles the widely used *frame/non-frame* categorization [Lifschitz, 1990] in the context of the Ramification Problem [Ginsberg and Smith, 1988a], which is employed to distinguish caused from unmotivated indirect effects. The attraction of this categorising events as either natural or volitional actions lies in its being straightforwardly realisable. By employing this minimization strategy a number of problems are resolved, including our introductory ones. Yet it is still crucial that the domain being modeled meets certain assumptions regarding mutual independence of events, namely, that natural events do not interfere and that actions are necessarily causally independent.

In Section 3, we assess these assumptions by presenting a scenario which involves only natural events and yet where some but not all events may be caused by others. Furthermore, we argue that even from the commonsense point of view actions are not necessarily causeless, namely, in case they are reflexive (hence not volitional), or in case of normative rules, stating that an action ought to be performed under certain circumstances. A universally applicable event minimization strategy therefore requires the incorporation of a suitable notion of causality by which it is generally possible to tell apart caused from unmotivated event occurrences.

Actually this general problem shows a striking similarity to the necessity of distinguishing caused from unmotivated indirect effects as part of the broader Ramification Problem. Research in this context has recently produced several successful approaches which appeal to causality (e.g., [Elkan, 1992; Geffner and Pearl, 1992; Lin, 1995; McCain and Turner, 1995; Thielscher, 1997]). In Section 4, we show how these results can be exploited to successfully address the problem considered in the present paper. The conceptually crucial step towards this end is to identify event occurrences with fluents. This allows us to interpret formulas like (2) as so-called state constraints, which then give rise to indirect effects. The problem of deriving the right event occurrences thus becomes part of the Ramification Problem, and so we can adapt, for instance, our causality-based approach of [Thielscher, 1997] to provide a solution. We present a high-level narrative description language in which is realized this solution to the event minimization problem. In Section 5, we furthermore illustrate how this solution can be axiomatized by a novel use of Fluent Calculus techniques. We conclude in Section 6.

2 ACTIONS VERSUS NATURAL EVENTS

The reason for global minimization failing to produce the intended conclusions in the introductory examples becomes apparent when we analyze the specifications (1) and (2) from a purely logical perspective. The constraint in (2), for instance, logically entails this formula:

$$\begin{aligned} Happens(\mathtt{lleft}, t) \wedge \neg Happens(\mathtt{spills}, t) \\ \supset Happens(\mathtt{lright}, t) \end{aligned}$$

This explains the existence of the unintended model, i.e., where $\neg Happens(\mathtt{spills}, 2)$ is assumed and $Happens(\mathtt{lright}, 2)$ follows. Similarly, formula (1) is equivalent to the implication,

$$\begin{aligned} Happens(\mathtt{walk}, t) \wedge \neg Happens(\mathtt{bump}, t+2) \\ \supset Happens(\mathtt{stop}, t+1) \end{aligned}$$

sanctioning the unintended model obtained in Example 1.

[2]It is worth mentioning that the more sophisticated so-called *chronological* minimisation [Shoham, 1988] would solve our first but not the second example, because the two conflicting events, $Happens(\mathtt{lright}, 2)$ vs. $Happens(\mathtt{spills}, 2)$, are in no chronological order. Likewise, in the approach of [Stein and Morgenstern, 1994] an event can only *motivate*, as it is called, another event if the latter occurs at a later time-point, i.e., not concurrently. So this notion of motivation, too, does not help us detecting the unintended model in Example 2.

These observations indicate that both the two specifications (1) and (2) just contain insufficient information to rule out the respective anomalous model. In fact, the reason for us rejecting some of the minimal models is that we employ additional domain knowledge to the effect that some events, like the spilling of the soup, may be the natural consequence of certain circumstances while other events, like the lifting of the table, require a volitional act. In other words, the occurrence of the latter depends on a deliberate, free-will decision of some agent. Whenever a conflict needs to be resolved, we seem to prefer the occurrence of a natural event rather than postulating the performance of an action which the narrative does not necessitate. This argument also applies to Example 1: The robot bumping at the wall is a natural consequence of its walking towards it, whereas the robot stopping beforehand involves the explicit decision to act.

The distinction between actions on the one hand and natural events on the other, provides the basis for a refined strategy for event minimization in case causal dependencies are to be taken into account. If a domain description includes knowledge as to the category to which each event belongs, then minimization can exploit this knowledge in order to rule out any unmotivated action. What is especially appealing is that this solution to our two introductory example problems can be straightforwardly realised, e.g. in the approach of [Shanahan, 1996] as follows: We first introduce two predicates $A\_Happens(e,t)$ and $E\_Happens(e,t)$. The former is to be used whenever e is an action, the latter whenever e is a natural event. Let N be a set of formulas formalizing the course of events in a narrative, then by using *priority circumscription* [Lifschitz, 1987], written $CIRC[N; A\_Happens > E\_Happens]$, $A\_Happens$ can be minimized with higher priority than $E\_Happens$.[3] This circumscription policy reflects our intention to prefer minimization of action performances over the occurrence of natural events.

Example 1 (continued) In accordance with the above proposal, we rewrite (1) as follows:

$$A\_Happens(\mathtt{walk},t) \wedge \neg A\_Happens(\mathtt{stop},t+1) \supset E\_Happens(\mathrm{bump},t+2)$$

which reflects the fact that both walking and stopping are actions while the robot's bumping at the wall is a natural event. Suppose again that the robot walks towards the wall at time $t = 5$, which is now represented by the fact $A\_Happens(\mathtt{walk},5)$. Then our refined circumscription policy yields the intended conclusion that $\neg A\_Happens(\mathtt{stop},6)$ and, hence, $E\_Happens(\mathrm{bump},7)$. ■

[3]In order that the various foundational axioms which involve *Happens* need not be rewritten, the formula $Happens(e,t) \equiv A\_Happens(e,t) \vee E\_Happens(e,t)$ should be added after circumscribing, in N, the new predicates.

Example 2 (continued) We rewrite (2) as follows:

$$A\_Happens(\mathtt{lleft},t) \equiv \neg A\_Happens(\mathtt{lright},t) \supset E\_Happens(\mathtt{spills},t)$$

Suppose again that the left hand side is lifted at time $t = 2$, now represented by $A\_Happens(\mathtt{lleft},2)$. As above, our refined circumscription policy yields the intended conclusion that $\neg A\_Happens(\mathtt{lright},2)$ and, hence, $E\_Happens(\mathtt{spills},2)$. ■

3 INTERACTING EVENTS, REFLEXIVE ACTIONS, AND NORMATIVE RULES

Categorizing the underlying events as either actions or natural events provides a minimization strategy sophisticated enough for both our two example scenarios discussed in the introduction. As regards the general range of applicability, a closer examination reveals two fundamental assumptions which an application domain needs to satisfy in order for this strategy to guarantee the intended results. First, there must never be a priority between two natural events unless one of them, but not the other, is triggered by an action that is known to take place. Second, actions themselves need always be independent both of each other and of natural events. That is to say, an action is never caused by the performance of another action nor by the occurrence of a natural event or any other circumstances. As long as a domain complies with these assumptions, the distinction between volitional actions and natural events provides a suitable minimization policy. We again stress that the value of this strategy lies in its being amenable to straightforward integration into existing approaches.

On the other hand, the aforementioned basic assumptions are not of universal nature. Consider, as an example for interfering natural events, the three events `tile_falls` (a tile falls from the roof), `tile_hits_plant` (the tile breaks a rare plant in the yard), and `tree_falls` (a tree falls down). Suppose the falling tile hits the plant unless, curiously enough, the tree falls down at the very same time, crossing the trajectory of the tile:

$$E\_Happens(\mathtt{tile\_falls},t) \wedge \neg E\_Happens(\mathtt{tree\_falls},t) \supset E\_Happens(\mathtt{tile\_hits\_plant},t)$$

What should be concluded e.g. from the sole fact that $E\_Happens(\mathtt{tile\_falls},2)$? Obviously, the tile hitting the plant is much more likely in this case than the tree falling at the very time. Since, however, all three involved events are natural, categorization-based minimization does not arrive at this conclusion. Rather it produces an additional model where $\neg E\_Happens(\mathtt{tile\_hits\_plant},2)$ and, consequently, $E\_Happens(\mathtt{tree\_falls},2)$ hold. This illustrates that natural events may interfere without the explicit performance of any action.

Regarding the second requirement for the applicability of the categorisation-based approach, consider, e.g., actions which are *reflexive*. These may well be the natural consequence of events. If, for example, we touch the working hot plate of a stove, then this causes us to involuntarily withdraw our hand. Nonetheless this withdrawing our hand cannot in general be considered a natural event because it may of course be a volitional act in other situations. The assumption that actions are always independent both of other actions and of events in general, does also not apply in case one intends to reason with *normative rules*, stating that an action ought to be performed in specific situations. The design of a robot's behavior might be given as set of such rules. Then again categorization-based minimisation can no longer be guaranteed to distinguish action occurrences that are to be expected and those which a narrative does not necessitate.

4 EVENT MINIMIZATION AND CAUSALITY

Due to its formal elegance, categorization-based event minimisation can be easily realized in existing frameworks. We have seen how this strategy solves the problem of unintended models whenever the distinction between actions and natural events is guaranteed to always help telling apart events whose occurrence admits a causal explanation. The spilling of the soup in Example 2, for instance, is a natural event *caused* by the lifting of the table on the left hand side, while the latter does not have the 'capability' of causing the volitional action of lifting up the right hand side of the table. On the other hand, in the preceding section we have met examples where our principle of categorization turned out too weak to identify event occurrences which are caused. Much like we have just argued, we can say that the tile's falling from the roof may cause the event that the plant breaks but cannot cause the simultaneous falling of the tree. And yet the two latter events both are natural, hence indistinguishable by the categorisation principle. It therefore seems that a universal solution to the problem of unintended models resulting from event minimization needs to directly appeal to the notion of causality.

Fortunately, there is no need to start from scratch to this end. A ready approach is furnished by an existing solution to a related aspect of a different problem. Namely, the situation we arrived at shows a striking similarity to the problem of deriving undesired indirect effects in the context of the Ramification Problem [Ginsberg and Smith, 1988b]. Generally, indirect effects of actions, or of events, are not explicitly represented in some effect specification but follow from general laws, so-called *state constraints*, which describe certain state-independent relations among fluents.[4] A well-known challenge in this context is to select only the intended ones among all the potential indirect effects a state constraint suggests from a purely syntactic perspective. A categorization principle has been proposed as a solution to this problem, too, (see [Lifschitz, 1990]) and is widely used (e.g., [del Val and Shoham, 1993; Brewka and Hertzberg, 1993; Kartha and Lifschitz, 1994; Sandewall, 1995])—but it just as well turned out applicable to a certain extent only [Thielscher, 1997]. Recent research results on the Ramification Problem show how this limitations can be overcome by directly appealing to causality (e.g., [Elkan, 1992; Geffner and Pearl, 1992; Lin, 1995; McCain and Turner, 1995; Thielscher, 1997]). In the following, we exploit these results for the development of a causality-based solution to the problem of deriving the wrong event occurrences.

In order that this can be done, we first have to resolve an apparent fundamental difference between *events* on the one hand, and (indirect) effects—updates of value assignments to *fluents*—on the other hand. A small but conceptually crucial step achieves this: Event occurrences are identified as fluents. That is to say, each state of the world is also characterized by the events which currently happen, if any. By this we adapt the *actions-as-fluents* paradigm, which has been propagated, for instance, in [Lin and Shoham, 1992; Große, 1994; Thielscher, 1995]. To emphasize this shift, we will write event fluents using the symbol happens instead of *Happens*. Relations among events, like the one formalized in equation (2), can now be considered state constraints, which supposedly hold in all states, e.g.[5]

$$\texttt{happens(lleft)} \equiv \neg\texttt{happens(lright)} \supset \texttt{happens(spills)}$$

Becoming state constraints, relations among event occurrences may give rise to indirect effects. If, for instance, happens(lleft) becomes true, then this additionally causes the indirect effect that happens(spills) according to the aforementioned constraint. Of course we need means to avoid the alternative conclusion that this constraint instead triggers the indirect effect happens(lright). In this way the key problem of the present paper has become part of the Ramification Problem, and so is amenable to the existing causality-based techniques developed in that context. In particular, we can, and will, adopt our

[4]A fluent is an atomic property (of some object) which may change in the course of time.

[5]Relations among events which refer to different time-points, like the one formalised in equation (1), cannot be reformulated solely by means of state constraints but by combining those with effect propositions (see below). Event occurrences triggered by such relations are then obtained as a combination of indirect and so-called delayed effects.

theory of causal relationships [Thielscher, 1997] and combine it with the events-as-fluents paradigm. The resulting theory is first presented as a formal, high-level narrative description language in the spirit of $\mathcal{A}$ [Gelfond and Lifschitz, 1993] or $\mathcal{E}$ [Kakas and Miller, 1997] etc. Thereafter, in Section 5, we show how this theory can be axiomatized on the basis of a novel variant of the Fluent Calculus.

Fluents are the only basic entity of domain descriptions in our language; events, and hence actions, are just a particular kind of fluent. As opposed to most 'ordinary' ones, event fluents have, however, a special characteristics: They are not subject to the commonsense law of persistence, which states that usually the value of a fluent 'tends to persist,' i.e., is stable unless an actual event effects a change. Event fluents are different in that they 'tend to disappear.' In this way event occurrences are always forced to have a cause. Following standard terminology, we call *momentary* all fluents of the latter kind, as proposed in [Lifschitz and Rabinov, 1989]. The other fluents we call *static*. Each momentary fluent has a designated default value, which it takes on unless, for an instant, something causes a different value.[6]

A *state* is composed of assignments of values to fluents. The atomic statement that a certain fluent f is of value v, written $f \hat{=} v$, is called a *fluent expression*. If f is a binary fluent with the domain $\{\texttt{false}, \texttt{true}\}$, then we abbreviate $f \hat{=} \texttt{false}$ by $\neg f$ and $f \hat{=} \texttt{true}$ simply by f. One more notation will be useful for later purpose: If S is a set of fluent assignments, then by $S|_{f \hat{=} v}$ we denote the set which is identical to S except for fluent f possessing value v.

Fluent expressions can be considered the underlying atoms for constructing *fluent formulas* using the standard logical connectives. The notion of fluent formulas being *true* in a state S is then based on defining a fluent expression $f \hat{=} v$ to be true if and only if $f \hat{=} v \in S$. *State constraints* are fluent formulas which have to be satisfied in all states that are possible in a domain.

Domain descriptions include propositions which specify the direct and indirect effects of events. The former are given by *effect propositions*, which indicate circumstances under which certain fluent changes are effected when moving on from one state to the next. An example is the effect proposition

$$\texttt{happens(spills)} \land \texttt{stain} \hat{=} \texttt{small} \\ \underline{\text{effects}}\ \texttt{stain} \hat{=} \texttt{large}$$

meaning that if soup spills out with the tablecloth already being lightly stained, then the situation gets worse during the next state transition. Our language supports the specification of non-deterministic effects, as in

$$\texttt{happens(spills)} \land \texttt{stain} \hat{=} \texttt{none} \\ \underline{\text{effects}}\ \texttt{stain} \hat{=} \texttt{small} \,|\, \texttt{stain} \hat{=} \texttt{large}$$

meaning that the spilling soup will produce either a small or large stain on a clean tablecloth.

Indirect effects of events are described by *causal relationships* [Thielscher, 1997], such as

$$\texttt{happens(lleft)}\ \underline{\text{causes}}\ \texttt{happens(spills)} \\ \underline{\text{if}}\ \neg\texttt{happens(lright)}$$

which indicates that if happens(lleft) occurs as direct or indirect effect of a state transition, then this triggers the additional, indirect effect happens(spills), provided ¬happens(lright) holds. Notice a crucial difference between the specifications of direct and indirect effects: Cause and effect in effect propositions refer to two consecutive states, and to one and the same state in causal relationships.

Definition 1 A *domain description* consists of

1. a set $\mathcal{F} = \mathcal{F}_s \cup \mathcal{F}_m$ of *static* and *momentary fluents*, each of which is associated with a non-empty *domain*;
2. a set of *effect propositions* of the form
$$C\ \underline{\text{effects}}\ E_1 \,|\, \dots \,|\, E_n$$
where the *condition* C is a fluent formula and the (alternative) *effects* E_i all are finite (possibly empty) sequences of fluent expressions ($n > 0$);[7]
3. a set of *causal relationships* of the form
$$\varepsilon\ \underline{\text{causes}}\ \varrho\ \underline{\text{if}}\ \Phi$$
where ε, ϱ (the *effect* and the *ramification*, respectively) are fluent expressions and the *context* Φ is a fluent formula;
4. a set $\mathcal{C}$ of fluent formulas, called the *state constraints*.

A *state* is a set of assignments $f \hat{=} v$ such that to each $f \in \mathcal{F}$ is assigned a value of its domain. A state S *satisfies* $\mathcal{C}$ iff each constraint is true in S. ■

Intuitively, an effect proposition C <u>effects</u> $E_1|\dots|E_n$ means that if C holds in some state, then exactly one of $E_1, \dots, E_n$ materializes in the next state. Thus disjunction is interpreted as exclusive; inclusive disjunction can of course be obtained by simply moving from, say, two alternatives $e_1 \,|\, e_2$ to three, viz. $e_1 \,|\, e_2 \,|\, e_1, e_2$. The intuitive meaning of a causal relationship ε <u>causes</u> ϱ <u>if</u> Φ is that if ε occurs as (direct

[6]In what follows, we assume for the sake of uniformity that all momentary fluents be binary with the value domain {false, true} and false being the default. This is just for the sake of clarity of presentation.

[7]If some E_i is empty, then one possibility for what happens if C holds is—nothing.

or indirect) effect and context Φ holds, then the indirect effect ϱ is additionally obtained. For causal relationships whose context is a logical tautology we use the short-hand form ε causes ϱ.[8]

Example 2 (continued) Consider the two static fluents $\mathcal{F}_s = \{\texttt{clock}, \texttt{stain}\}$ with the natural numbers and $\{\texttt{none}, \texttt{small}, \texttt{large}\}$ as the respective domain (where the latter indicates how badly the tablecloth is stained, if at all). Consider further the three momentary (hence binary) fluents $\mathcal{F}_m = \{\texttt{happens}(e) : e \in \{\texttt{lleft}, \texttt{lright}, \texttt{spills}\}\}$. Then the following components together constitute a domain description.

- The state constraint,

$$\begin{array}{r}\texttt{happens(lleft)} \equiv \neg\texttt{happens(lright)} \\ \supset \texttt{happens(spills)}\end{array} \qquad (3)$$

- The corresponding causal relationships,

$$\begin{array}{rl}\texttt{happens(lleft)} \ \underline{\text{causes}} & \texttt{happens(spills)} \\ & \underline{\text{if}}\ \neg\texttt{happens(lright)} \\ \texttt{happens(lright)} \ \underline{\text{causes}} & \texttt{happens(spills)} \\ & \underline{\text{if}}\ \neg\texttt{happens(lleft)}\end{array} \qquad (4)$$

- The effect propositions,

$$\begin{array}{l}\texttt{happens(spills)} \wedge \texttt{stain}\hat{=}\texttt{none} \\ \quad \underline{\text{effects}}\ \texttt{stain}\hat{=}\texttt{small} \mid \texttt{stain}\hat{=}\texttt{large} \\ \texttt{happens(spills)} \wedge \neg\texttt{stain}\hat{=}\texttt{none} \\ \quad \underline{\text{effects}}\ \texttt{stain}\hat{=}\texttt{large} \\ \texttt{clock}\hat{=}t\ \underline{\text{effects}}\ \texttt{clock}\hat{=}t+1 \\ \texttt{clock}\hat{=}1\ \underline{\text{effects}}\ \texttt{happens(lleft)}\end{array} \qquad (5)$$

The last but one effect proposition is a representative of all instances where t is a natural number. The very last proposition formalises the initiation of a `lleft` action at time 2.

■

The effect proposition in (5) with condition $\texttt{clock}\hat{=}1$ illustrates how causal chains of events are initiated: Volitional actions or natural events which common sense prefers to consider causeless[9] are caused just by reaching the time-point at which they occur. Obviously, the fluent `clock` is crucial for this purpose, and so we assume it to be contained in any domain description. The time structure underlying our approach is left-bound, linear, and discrete; a challenge for future research is the generalization to continuous time and change.

Next we define the notion of successor states, which are obtained according to underlying effect laws and causal relationships. Since we allow for non-determinism, a state may admit several possible successors. We begin by defining so-called *preliminary* successor states, in which all direct effects have been accounted for but which require further investigation to accommodate possible indirect effects.

Definition 2 Consider a domain description with static and momentary fluents $\mathcal{F}_s$ and $\mathcal{F}_m$, respectively; effect propositions $\mathcal{E}$; and state constraints $\mathcal{C}$. Let S be a state satisfying $\mathcal{C}$, then any state S' which obtains as follows is called a *preliminary successor of* S: For each C $\underline{\text{effects}}$ $E_1|\ldots|E_n \in \mathcal{E}$ such that C holds in S, select one E_i $(1 \leq i \leq n)$. Let E be the entire set of assignments thus obtained. Then S' consists of

1. all assignments in E,
2. all assignments $f\hat{=}\texttt{false}$ for $f \in \mathcal{F}_m$ such that E contains no assignment for f,
3. all assignments $f\hat{=}v \in S$ for $f \in \mathcal{F}_s$ such that E contains no assignment for f.

■

Put in words, the preliminary successors are obtained by first making a selection among the alternative outcomes of all applicable effect propositions and, then, by realizing all these effects, by setting all unaffected momentary fluents to their default value, and by letting all unaffected static fluents persist.[10]

Example 2 (continued) Consider this state:

$$\begin{array}{l}S(1) = \{\neg\texttt{happens(lleft)}, \neg\texttt{happens(lright)}, \\ \qquad \texttt{happens(spills)}, \texttt{clock}\hat{=}1, \texttt{stain}\hat{=}\texttt{none}\}\end{array}$$

Of the effect propositions in (5) three are applicable,

[8]Causal relationships are typically rooted in state constraints but provide additional causal information. In [Thielscher, 1997] we have shown that it is not necessary to draw up causal relationships by hand. Rather these can be fully automatically extracted from a given set of state constraints plus suitable knowledge as to which fluents have the potential to causally affect what other fluents. Later on we will discuss this point in greater detail.

[9]While the universal causal law stipulates that everything has a cause, common sense always considers only a fraction of the whole universe and, hence, only a fraction of the entire history of events. Anything whose cause lies outside this fraction is considered causeless.

[10]It is noteworthy that in case of contradictory effect propositions the resulting set of assignments may be inconsistent in that it contains some $f\hat{=}v_1$ together with $f\hat{=}v_2$ such that $v_1 \neq v_2$. Then this set is not a state, hence does not constitute a preliminary successor according to the definition. States therefore may admit no preliminary and also no successor states at all.

namely,

$$\begin{array}{l}\texttt{happens(spills)} \wedge \texttt{stain} \hat{=} \texttt{none} \\ \quad \underline{\text{effects}}\ \texttt{stain} \hat{=} \texttt{small} \mid \texttt{stain} \hat{=} \texttt{large} \\ \texttt{clock} \hat{=} 1\ \underline{\text{effects}}\ \texttt{clock} \hat{=} 2 \\ \texttt{clock} \hat{=} 1\ \underline{\text{effects}}\ \texttt{happens(lleft)}\end{array}$$

The topmost proposition being indeterminate, we obtain two preliminary successor states, viz.

$$\begin{array}{l}\{\ \texttt{happens(lleft)}, \neg\texttt{happens(lright)}, \\ \quad \neg\texttt{happens(spills)}, \texttt{clock} \hat{=} 2, \texttt{stain} \hat{=} \texttt{small}\ \} \\ \{\ \texttt{happens(lleft)}, \neg\texttt{happens(lright)}, \\ \quad \neg\texttt{happens(spills)}, \texttt{clock} \hat{=} 2, \texttt{stain} \hat{=} \texttt{large}\ \}\end{array}$$

Notice how the momentary fluent **happens(spills)** is automatically set to its default value, **false**. Notice further that neither of the two preliminary successors satisfies the state constraint (3), for we have not yet considered the possibility of indirect effects. ■

In order to account for indirect effects, preliminary successors are taken as starting points for 'causal propagation,' that is the successive application of causal relationships until overall satisfactory successor states obtain [Thielscher, 1997]. Formally, causal relationships operate on pairs (S, E), where S denotes an intermediate state, in which some but not yet all indirect effects have been realized, and where E contains all direct and indirect effects computed so far:[11]

Definition 3 Consider a pair (S, E) consisting of a state S and a set of fluent expressions E. A causal relationship ε causes ϱ if Φ is *applicable* to (S, E) iff $\Phi \wedge \neg\varrho$ is true in S and $\varepsilon \in E$. Its *application* yields the pair $(S|_\varrho, E|_\varrho)$. ■

That is to say, a causal relationship is applicable if the associated condition Φ holds, the particular indirect effect ϱ is currently false, and its cause ε is among the current effects. If $\mathcal{R}$ is a set of causal relationships, then by $(S, E) \stackrel{*}{\leadsto}_{\mathcal{R}} (S', E')$ we indicate that there is a (possibly empty) sequence of elements of $\mathcal{R}$ so that the successive application to (S, E) results in (S', E'). It is easy to verify that if S is a state and E is consistent (i.e., contains no double assignments to fluents), then $(S, E) \stackrel{*}{\leadsto}_{\mathcal{R}} (S', E')$ implies that S' is a state and E' is consistent, too.

Now suppose given a set of fluent expressions S as the result of having accounted for all direct effects E via the given effect propositions. This preliminary successor S may violate the state constraints. Additional, indirect effects are then accommodated by (non-deterministically) selecting and (serially) applying causal relationships until a state satisfying all state constraints obtains.

[11] For a clarification of the crucial role of the second component, E, as well as for further details we suggest to consult [Thielscher, 1997].

Definition 4 Consider a domain description with causal relationships $\mathcal{R}$ and state constraints $\mathcal{C}$. Furthermore, let S be a state satisfying $\mathcal{C}$. A state T is a *possible successor state* of S iff there exists a preliminary successor S' obtained through direct effects E and such that

1. $(S', E) \stackrel{*}{\leadsto}_{\mathcal{R}} (T, E')$ for some E', and
2. T satisfies $\mathcal{C}$. ■

Example 2 (continued) Starting off from the two preliminary successors of our state $S(1)$ from above, on the basis of the causal relationships and state constraints (4) and (3), respectively, we obtain two possible successor states, viz.

$$\begin{array}{l}\{\ \texttt{happens(lleft)}, \neg\texttt{happens(lright)}, \\ \quad \texttt{happens(spills)}, \texttt{clock} \hat{=} 2, \texttt{stain} \hat{=} \texttt{small}\ \} \\ \{\ \texttt{happens(lleft)}, \neg\texttt{happens(lright)}, \\ \quad \texttt{happens(spills)}, \texttt{clock} \hat{=} 2, \texttt{stain} \hat{=} \texttt{large}\ \}\end{array}$$

Both these two successor states are obtained by application of the causal relationship

$$\begin{array}{l}\texttt{happens(lleft)}\ \underline{\text{causes}}\ \texttt{happens(spills)} \\ \qquad\qquad\qquad \underline{\text{if}}\ \neg\texttt{happens(lright)}\end{array}$$

It applies as $\neg$**happens(lright)** $\wedge$ $\neg$**happens(spills)** holds in the respective preliminary successor state, and on account of **happens(lleft)** being among the direct effects. ■

Obtaining the intended result by applying causal relationships to accommodate indirect effects depends, to state the obvious, on a suitable set of these relationships. This set should be complete in that it covers all indirect effects that reasonably follow from the state constraints, and, in particular, it should be sound in that it does not sanction indirect effects which do not follow from the standpoint of causality. Our two causal relationships of equation (4), for instance, constitute such a suitable set. Notice, however, that from a purely syntactical point of view state constraint (3) suggests additional, unintended causal relationships, such as

$$\begin{array}{l}\texttt{happens(lleft)}\ \underline{\text{causes}}\ \texttt{happens(lright)} \\ \qquad\qquad\qquad \underline{\text{if}}\ \neg\texttt{happens(spills)}\end{array}$$

Precisely this is the motivation for employing causal relationships, which convey more information than the mere state constraints. In [Thielscher, 1997] we have argued that causal relationships need not be drawn up all by hand but can be automatically generated on the basis of additional domain knowledge as to potential causal influence of some fluents upon others. In its simplest form, this knowledge is formally provided by a binary relation $\mathcal{I}$ on fluents, called *influence information*. If $(f_1, f_2) \in \mathcal{I}$, then this is intended to denote that a change of f_1's value potentially causally affects the value of f_2. In our running example, the suitable influence information is to let $\mathcal{I}$ consist of the two elements (**happens**(`lleft`), **happens**(`spills`)) and (**happens**(`lright`), **happens**(`spills`)). That is to say, the events `lleft` and `lright` may causally affect the event `spills` but not vice versa, nor do they mutually interfere. If applied to the state constraint of equation (3), the two causal relationships (4) are obtained. For further details we refer the reader to [Thielscher, 1997].[12]

The formal definition of successor states completes the crucial part of our narrative description language. What remains to be done is to introduce *observation* statements and, then, to give a precise notion of models, which are *histories*, and of entailment. Reflecting the intention to never consider event occurrences without a cause, we require that each momentary fluent takes on its default value at the initiation of a history. In so doing we employ the general principle of initial minimisation (see, e.g., [Shanahan, 1995b; Thielscher, 1996]). It is further assumed that the clock always shows the right time.

Definition 5 A *formal narrative* is a domain description augmented by a set of *observations*, which are expressions of the form $[t]\,F$ where t is a timepoint and F a fluent formula. A *history* is an infinite sequence of states $S(0), S(1), S(2), \ldots$ Such a history is a *model* of a formal narrative iff

1. each momentary fluent is false in $S(0)$,
2. each $S(t+1)$ is a successor state of $S(t)$ $(t \geq 0)$,
3. $\mathtt{clock}\hat{=}t \in S(t)$ for each $t \geq 0$, and
4. for each observation $[t]\,F$, fluent formula F is true in state $S(t)$.

An observation is *entailed* iff it holds in all models.

Example 2 (continued) Let us add to our example domain description the effect proposition

$$\mathtt{clock}\hat{=}0 \ \underline{\text{effects}}\ \mathbf{happens}(\mathtt{spills}) \tag{6}$$

Consider the narrative consisting of the resulting domain description plus the observation that initially the tablecloth is not badly stained, i.e.,

$$[0]\,\neg\mathtt{stain}\hat{=}\mathtt{large} \tag{7}$$

Then any model must have initial state

$$\begin{aligned} S(0) = \{\, &\neg\mathbf{happens}(\mathtt{lleft}), \neg\mathbf{happens}(\mathtt{lright}), \\ &\neg\mathbf{happens}(\mathtt{spills}), \mathtt{clock}\hat{=}0, \mathtt{stain}\hat{=}v \} \end{aligned} \tag{8}$$

where v is either `none` or `small`. Following (6), a `spills` event occurs at time $t = 1$. According to the topmost two effect propositions in (5), this event in turn produces a small or a large stain in case $\mathtt{stain}\hat{=}\mathtt{none} \in S(0)$, and a large stain in case $\mathtt{stain}\hat{=}\mathtt{small} \in S(0)$. Hence the narrative entails

$$[2]\,\mathtt{stain}\hat{=}\mathtt{small} \vee \mathtt{stain}\hat{=}\mathtt{large} \tag{9}$$

Now, according to the bottommost effect proposition in (5), a `lleft` event occurs at the same time, which, as we have seen, has the indirect effect of yet another `spills` event. Therefore, knowing that (9) holds in all models, we see that our narrative also entails

$$[3]\,\mathtt{stain}\hat{=}\mathtt{large}$$

■

As an important feature our notion of entailment supports explanatory reasoning, i.e., reasoning backwards in time, as far as incomplete knowledge of static fluents is concerned. For instance, if the observation $[2]\,\mathtt{stain}\hat{=}\mathtt{small}$ were added to the example narrative from above, then it is easy to see that the observation $[0]\,\mathtt{stain}\hat{=}\mathtt{none}$ would be entailed.

What cannot be plainly derived from a narrative are events which are not caused by what is known to happen. For instance, if the observation $[2]\,\neg\mathbf{happens}(\mathtt{spills})$ were added to our example narrative, then the latter would admit no models at all because the observation cannot be explained without granting a new causeless event. Additional means are needed to abduce a suitable explanation, e.g., that a `lright` event occurs at time $t = 2$ in addition to the `lleft` event.

This property of our high-level language and semantics carries over to the axiomatization to be presented in the next section. If it concerns values of static fluents, an explanation for observed facts will be deductively derivable. Abduction will be required to explain observations by uncaused event happenings.

5 A FLUENT CALCULUS AXIOMATIZATION

Having presented a high-level language for describing and reasoning about narratives, we will now illustrate a way of axiomatising narratives described in this language so that our specific notion of entailment becomes entailment in classical logic and, hence, the reasoning can be carried out by fully automated deduction. Just like our narrative description language does it, the resulting axiomatization successfully copes with the problem of causally connected events. We restrict ourselves to deterministic domains, i.e., where all states admit a unique successor state. Due to lack of space, we illustrate the axiomatization

[12]The method described in [Thielscher, 1997] is originally defined only for state constraints over two-valued fluents, but the generalisation is straightforward. An unsatisfactory property is that this method may yield different sets of causal relationships for semantically equivalent state constraints. Following a suggestion by Javier Pinto, independence of syntax is achieved by processing the prime implicants of a set of constraints.

merely by example. General correctness wrt. the notion of entailment in our high-level language is proved in [Thielscher, 1998a].

Our axiomatisation is based on a novel use of Fluent Calculus, which was introduced in [Hölldobler and Schneeberger, 1990] and so christened in [Bornscheuer and Thielscher, 1997]. While historically the Fluent Calculus arose from approaches to the Frame Problem using non-classical, linear logics, in [Thielscher, 1998b] we argue that it can alternatively be viewed as a development of the Situation Calculus in order to cope with both the representational and the inferential aspect of the Frame Problem, without leaving classical logic. The key to this new Fluent Calculus is to reformulate successor state axioms [Reiter, 1991] applying the principle of reification, which means to use terms instead of atoms as the formal denotation of statements. To be more specific, reification in the Fluent Calculus means not only to denote single fluent-value assignments $f \hat{=} v$ as terms (which we will write as $\langle f, v \rangle$), but also conjunctions of them. Required to this end is a binary function, denoted by the symbol "$\circ$" and written in infix notation, by which conjunction is reified. The great advantage of so doing is that term variables can occur in state descriptions to indicate incomplete knowledge of the state at hand, as in, e.g., the specification

$$\exists z, v\, [\, S_0 = \langle \mathtt{clock}, 0 \rangle \circ \langle \mathtt{stain}, v \rangle \circ z \,\wedge\, v \neq \mathtt{large}\,]$$

which says that of state S_0 it is merely known that $\mathtt{clock} \hat{=} 0$ holds and that $\mathtt{stain} \hat{=} \mathtt{large}$ is false.

Central to the Fluent Calculus variant of [Thielscher, 1998b] is a function $State(s)$ which assigns to each situation a state term. This allows to rewrite successor state axioms to so-called *state update axioms*. These are of the form $\Delta[s] \supset \Gamma[State(Do(a, s)), State(s)]$, where $\Delta[s]$ describes the conditions on situation s under which the state associated with s is updated according to Γ to become the state associated with situation $Do(a, s)$.

A most interesting feature of this new Fluent Calculus is that by a slight modification it can be adapted from its branching time structure, which is typical for the Situation Calculus, to linear time, which brings it closer to the Event Calculus. The basic idea is to let the function $State$ range over time-points instead of situations. For instance, the following is an assertion about the initial state by which is axiomatised observation (7) of our example narrative at the end of the preceding section:

$$\exists z, v\, [\, State(0) = \langle \mathtt{stain}, v \rangle \circ z \,\wedge\, v \neq \mathtt{large}\,]$$

The binary function which reifies the logical conjunction needs to inherit from the latter an important property. In logical conjunctions the order is irrelevant in which the elements are given. Formally, order ignorance is ensured by stipulating the laws of associativity and commutativity, that is,

$$\begin{aligned} \forall x, y, z.\ (x \circ y) \circ z &= x \circ (y \circ z) \\ \forall x, y.\qquad x \circ y &= y \circ x \end{aligned}$$

It is convenient to also reify the empty conjunction, a logical tautology, by a constant denoted $\emptyset$ and which satisfies

$$\forall x.\ x \circ \emptyset = x$$

The three equational axioms, jointly abbreviated AC1, in conjunction with the standard axioms of equality entail the equivalence of two state terms whenever they are built up from an identical collection of reified fluents.[13]

A new feature required for our solution to the event minimization problem is the notion of momentary fluents. These are declared using a unary predicate $Momentary(f)$ in conjunction with a binary predicate $DefaultValue(f, v)$ determining the default value v of fluent f. A third predicate, $InDomain(f, v)$, is used to specify domains for the fluents. For our running example, adequate definitions of these three predicates are the following:[14]

$$\begin{aligned} Momentary(f) &\equiv f = \mathbf{happens(e)} \\ DefaultValue(f, v) &\equiv f = \mathbf{happens(e)} \wedge v = \mathtt{false} \\ InDomain(f, v) & \\ \equiv f = \mathbf{happens(e)} &\wedge (v = \mathtt{true} \vee v = \mathtt{false}) \\ \vee\ f = \mathtt{clock} &\wedge \exists t.\ v = t \\ \vee\ f = \mathtt{stain} &\wedge \\ (v = \mathtt{none} \vee v &= \mathtt{small} \vee v = \mathtt{large}) \end{aligned}$$

The above being domain-dependent axioms, we now introduce two foundational axioms which define the space of possible states. First, in a state to each fluent must be assigned a value of its domain:

$$\begin{aligned} &\exists v.\ InDomain(f, v) \\ &\quad \supset \exists z, v'\, [\, State(t) = \langle f, v' \rangle \circ z \,\wedge\, InDomain(f, v')\,] \end{aligned}$$

Second, no two values shall be assigned to the same fluent:

$$State(t) \neq \langle f, v \rangle \circ \langle f, v' \rangle \circ z$$

In order to increase readability of statements about states, we introduce a predicate $Holds(f, v, s)$ as an

[13]The reader may wonder why function $\circ$ is not expected to be idempotent, i.e., $\forall x.\ x \circ x = x$, which is yet another property of logical conjunction. The (subtle) reason for this is given below.

[14]In what follows, variables will be denoted by (possibly primed) lower-case letters. The particular variable t shall range over the natural numbers, including 0. Free variables in formulas are assumed universally quantified. For the sake of readability, we will furthermore use a variable $\mathbf{e}$ which can be replaced by either of our three events $\mathtt{lleft}$, $\mathtt{lright}$, or $\mathtt{spills}$.

abbreviation, meaning that fluent f has value v in state s:

$$Holds(f, v, s) \equiv \exists z.\, s = \langle f, v\rangle \circ z$$

A simple use of this macro is to ascertain that time and the value of the fluent clock coincide:

$$Holds(\texttt{clock}, t, State(t))$$

Effect propositions are axiomatised by implications of the form $\Delta[s] \supset Effects(f, v, v', s)$, where $\Delta[s]$ is a specification of the conditions that must be satisfied in state s in order that fluent f changes its value from v to v'. For example, the three effect propositions

$$\begin{array}{rll} \texttt{happens(spills)} & \underline{\text{effects}} & \texttt{stain}\hat{=}\texttt{large} \\ \texttt{clock}\hat{=}t & \underline{\text{effects}} & \texttt{clock}\hat{=}t+1 \\ \texttt{clock}\hat{=}1 & \underline{\text{effects}} & \texttt{happens(lleft)} \end{array}$$

are axiomatised as follows:

$$Holds(\texttt{happens(spills)}, \texttt{true}, s) \wedge Holds(\texttt{stain}, v, s) \supset Effects(\texttt{stain}, v, \texttt{large}, s)$$

$$Holds(\texttt{clock}, t, s) \supset Effects(\texttt{clock}, t, t+1, s)$$

$$Holds(\texttt{clock}, 1, s) \wedge Holds(\texttt{happens(lleft)}, v, s) \supset Effects(\texttt{happens(lleft)}, v, \texttt{true}, s)$$

In addition, a foundational axiom ensures that each momentary fluent changes to its default value if currently it enjoys another value and if no effect proposition implies its getting a non-default value:

$$\begin{array}{c} Momentary(f) \wedge DefaultValue(f, v) \wedge Holds(f, v', s) \\ \wedge\, v \neq v' \wedge \neg(Effects(f, v', v'', s) \wedge v \neq v'') \\ \supset Effects(f, v', v, s) \end{array}$$

The effect propositions are assumed to constitute a complete description of what changes when moving from one state to the next. In order to reflect this assumption, we circumscribe the predicate *Effects*, and so solve the representational aspect of the Frame Problem. The crucial next step is to solve the inferential aspect, too.

Taken together, the single effects can be viewed as a sound and complete set of constraints on two terms τ and ι in which are conjoined, via "$\circ$", the assignments that terminate to hold and the assignments that are initiated, respectively. These terms are then employed for updating the current state so as to arrive at the successor. Updating means that all unaffected fluent-value pairs remain untouched in the new state. Thus the inferential Frame Problem gets solved. The definition for τ and ι is as follows:

$$Change(t, \tau, \iota) \equiv \left[\begin{array}{c} Effects(f, v, v', State(t)) \\ \equiv \exists z, z' \left[\begin{array}{l} \tau = \langle f, v\rangle \circ z \wedge \\ \iota = \langle f, v'\rangle \circ z' \end{array} \right] \end{array} \right]$$

If only direct effects were to be considered, then the concept of successor states could now be modeled by the elegant schematic implication $State(t) = \tau \circ z \supset State(t+1) = z \circ \iota$. Incidentally, this scheme is the reason for not stipulating that $\circ$ be idempotent. For if it were, then, given that $State(t)$ includes the fluent-value assignments τ, the equation $State(t) = \tau \circ z$ would be satisfied if z is substituted by $State(t)$. Hence equating $State(t+1)$ with $z \circ \iota$ would not guarantee that all assignments in τ terminate.

In addition to the direct effects, ramifications need to be accounted for. The definition of how to obtain successor states therefore employs the predicate $Ramify(s, e, s')$ as introduced in [Thielscher, 1997], which is meant to be true if the successive application of causal relationships to (S, E) eventually results in a pair whose first component, S', satisfies the domain constraints—where s, e, and s' are reifications of S, E, and S', respectively. With this predicate, to be defined below, the following foundational axiom models the definition of successor states according to Definition 4:

$$\begin{array}{c} Change(t, \tau, \iota) \wedge State(t) = \tau \circ z \wedge Ramify(z \circ \iota, \iota, s) \\ \supset State(t+1) = s \end{array}$$

The definition of *Ramify* can be directly adopted from [Thielscher, 1997] as being the transitive closure of a predicate $Causes(s, e, s', e')$, which in turn is meant to be true iff there is an instance of a causal relationship which is applicable to (S, E) and whose application yields (S', E')—where s, e, s', e' are reifications of S, E, S', E'. The causal relationships in our example domain, c.f. equation (4), are thus suitably axiomatised as follows:

$$\begin{array}{lll} \multicolumn{3}{l}{Causes(s, e, s', e')} \\ \equiv & & \exists x.\, e = \langle \texttt{happens(lleft)}, \texttt{true}\rangle \circ x \\ & & \wedge\, Holds(\texttt{happens(lright)}, \texttt{false}, s) \\ & & \wedge\, s = \langle \texttt{happens(spills)}, \texttt{false}\rangle \circ y \\ & & \wedge\, s' = y \circ \langle \texttt{happens(spills)}, \texttt{true}\rangle \\ & & \wedge\, e' = e \circ \langle \texttt{happens(spills)}, \texttt{true}\rangle \\ & \vee & \exists x.\, e = \langle \texttt{happens(lright)}, \texttt{true}\rangle \circ x \\ & & \wedge\, Holds(\texttt{happens(lleft)}, \texttt{false}, s) \\ & & \wedge\, s = \langle \texttt{happens(spills)}, \texttt{false}\rangle \circ y \\ & & \wedge\, s' = y \circ \langle \texttt{happens(spills)}, \texttt{true}\rangle \\ & & \wedge\, e' = e \circ \langle \texttt{happens(spills)}, \texttt{true}\rangle \end{array}$$

Transitive closure cannot be expressed in first-order logic, which is why predicate *Ramify* is defined using the standard way of encoding transitive closure by a second-order formula:

$$\begin{array}{l} Ramify(s, e, s') \equiv \\ Possible(s') \wedge \forall \Pi. \\ \left\{ \begin{array}{c} \forall s_1, e_1.\, \Pi(s_1, e_1, s_1, e_1) \\ \wedge \\ \left[\begin{array}{l} \forall s_1, e_1, s_2, e_2, s_3, e_3 \\ \quad [\Pi(s_1, e_1, s_2, e_2) \wedge Causes(s_2, e_2, s_3, e_3) \\ \quad\quad \supset \Pi(s_1, e_1, s_3, e_3)] \end{array} \right] \\ \supset \\ \Pi(s, e, s', e') \end{array} \right\} \end{array}$$

That is, $Ramify(s, e, s')$ is true iff s' satisfies the state constraints and there is some e' such that (s, e, s', e') belongs to the transitive closure of *Causes*.

What remains to be axiomatized are the state constraints of a domain description, which in our example looks as follows:

$$Possible(s) \equiv \left\{ \begin{array}{l} \left[\begin{array}{l} Holds(\texttt{happens(lleft)}, \texttt{true}, s) \\ \equiv Holds(\texttt{happens(lright)}, \texttt{false}, s) \end{array} \right] \\ \quad \supset Holds(\texttt{happens(spills)}, \texttt{true}, s) \end{array} \right\}$$

This completes our axiomatization. We refer to [Thielscher, 1998a] for a proof of general correctness—wrt. our narrative description language—of axiomatizations based on the ideas illustrated here. Let us just mention the following specific result:

Theorem 6 *Let* Σ *be the conjunction of all Fluent Calculus formulas above. Then* $CIRC[\Sigma; Effects]$ *entails*

$$Holds(\texttt{stain}, \texttt{large}, State(2))$$

6 CONCLUSION

Coming from the observation that straightforwardly minimizing events in narratives is insufficient in case of causal dependencies among events, we have proposed two refined minimization strategies. The first of which exploits the distinction between two categories of events, namely, volitional actions vs. natural events. The range of applicability of this approach has been discussed, and we have then developed a general solution which helps telling apart caused event occurrences by directly appealing to the notion of causality.

Our first, categorization-based minimization strategy is particularly appealing because it can be easily integrated into existing approaches, which we have illustrated with the Event Calculus axiomatization of [Shanahan, 1996]. For our second, more general strategy we have identified event occurrences with fluents, which then has allowed us to apply an existing causality-based solution to the Ramification Problem. We have presented a high-level narrative description language and a novel Fluent Calculus axiomatization in which is realized this solution to the event minimization problem.

Regarding related work, we first note that arguments in favor of the theory of causal relationships as a solution to the Ramification Problem, and a through comparison to other approaches, can be found in the article [Thielscher, 1997]. In particular, there we have argued that excluding uncaused indirect effects, and not minimizing change, is the real issue of dealing with ramifications. This is especially crucial in case of so-called stabilizing state constraints [Thielscher, 1998c]. As for comparisons with existing event minimization strategies, it has already been mentioned that *chronological minimization* [Shoham, 1988] is not applicable to specifications like (2), i.e., which involve concurrent events. This argument applies to *Motivated Action Theory* [Stein and Morgenstern, 1994], too.

Our novel Fluent Calculus appears to be related in several interesting respects to the Event Calculus, in particular to the variant of [Shanahan, 1995a], where an explicit notion of state is used. A detailed comparison of the two is an important aspect of ongoing research.

Finally, we should stress that reasonably minimizing events is of great importance not only for reasoning about narratives, where it is a mere convention that events do not happen unless they follow from what has been said. It is also essential for setting up plans: Suppose a planning goal be to produce a stain in the tablecloth. A reasonable plan would be, say, to lift up the left hand side of the table on which the bowl of soup is located. Yet this plan can be concluded successful only under the assumption that no intervening agent lifts the right hand side simultaneously. Generally, it is difficult if not impossible to devise plans that are perfectly reliable in reality. Assuming away disturbing events for which there is no indication that they will occur, is therefore the only way to come up with plans which at least by default can be concluded to achieve a goal. Of course things may not turn out as expected when a plan is being executed. Agents therefore need to constantly update, e.g. by abduction, their knowledge about the actual course of events.

References

[Bornscheuer and Thielscher, 1997] S.-E. Bornscheuer and M. Thielscher. Explicit and implicit indeterminism: Reasoning about uncertain and contradictory specifications of dynamic systems. *J. of Logic Programming*, 31(1–3):119–155, 1997.

[Brewka and Hertzberg, 1993] G. Brewka and J. Hertzberg. How to do things with worlds: On formalizing actions and plans. *J. of Logic and Computation*, 3(5):517–532, 1993.

[del Val and Shoham, 1993] A. del Val and Y. Shoham. Deriving properties of belief update from theories of action (II). In R. Bajcsy, editor, *Proceedings of the IJCAI*, pages 732–737, Chambéry, France, August 1993. Morgan Kaufmann.

[Elkan, 1992] C. Elkan. Reasoning about action in first-order logic. In *Proceedings of the Conference of the Canadian Society for Computational Studies of Intelligence (CSCSI)*, pages 221–227, Vancouver, Canada, May 1992. Morgan Kaufmann.

[Geffner and Pearl, 1992] H. Geffner and J. Pearl. Conditional entailment: bridging two approaches to default reasoning. *Artificial Intelligence*, 23:209–244, 1992.

[Gelfond and Lifschitz, 1993] M. Gelfond and V. Lifschitz. Representing action and change by logic programs. *J. of Logic Programming*, 17:301–321, 1993.

[Ginsberg and Smith, 1988a] M. L. Ginsberg and D. E. Smith. Reasoning about action I: A possible worlds approach. *Artificial Intelligence*, 35:165–195, 1988.

[Ginsberg and Smith, 1988b] M. L. Ginsberg and D. E. Smith. Reasoning about action II: The qualification problem. *Artificial Intelligence*, 35:311–342, 1988.

[Große, 1994] G. Große. Propositional State-Event Logic. In C. MacNish, D. Peirce, and L. M. Peireira, editors, *Proceedings of the European Workshop on Logics in AI (JELIA)*, volume 838, pages 316–331. Springer, September 1994.

[Hölldobler and Schneeberger, 1990] S. Hölldobler and J. Schneeberger. A new deductive approach to planning. *New Generation Computing*, 8:225–244, 1990.

[Kakas and Miller, 1997] A. Kakas and R. Miller. A simple declarative language for describing narratives with actions. *J. of Logic Programming*, 31(1–3):157–200, 1997.

[Kartha and Lifschitz, 1994] G. N. Kartha and V. Lifschitz. Actions with indirect effects. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Proceedings of the KR*, pages 341–350, Bonn, Germany, May 1994. Morgan Kaufmann.

[Lifschitz and Rabinov, 1989] V. Lifschitz and A. Rabinov. Things that change by themselves. In *Proceedings of the IJCAI*, pages 864–867, Detroit, MI, 1989.

[Lifschitz, 1987] V. Lifschitz. Pointwise circumscription. In M. L. Ginsberg, editor, *Readings in Nonmonotonic Reasoning*, chapter 3.3, pages 179–193. Morgan Kaufmann, 1987.

[Lifschitz, 1990] V. Lifschitz. Frames in the space of situations. *Artificial Intelligence*, 46:365–376, 1990.

[Lin and Shoham, 1992] F. Lin and Y. Shoham. Concurrent actions in the situation calculus. In *Proceedings of the AAAI*, pages 590–595, San Jose, CA, 1992. MIT Press.

[Lin, 1995] F. Lin. Embracing causality in specifying the indirect effects of actions. In C. S. Mellish, editor, *Proceedings of the IJCAI*, pages 1985–1991, Montreal, Canada, August 1995. Morgan Kaufmann.

[McCain and Turner, 1995] N. McCain and H. Turner. A causal theory of ramifications and qalifications. In C. S. Mellish, editor, *Proceedings of the IJCAI*, pages 1978–1984, Montreal, Canada, August 1995. Morgan Kaufmann.

[Reiter, 1991] R. Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation*, pages 359–380. Academic Press, 1991.

[Sandewall, 1995] E. Sandewall. Reasoning about actions and change with ramification. In *Computer Science Today*, volume 1000 of *LNCS*. Springer, 1995.

[Shanahan, 1995a] M. Shanahan. A circumscriptive calculus of events. *Artificial Intelligence*, 77:249–284, 1995.

[Shanahan, 1995b] M. Shanahan. Default reasoning about spatial occupancy. *Artificial Intelligence*, 74:147–163, 1995.

[Shanahan, 1996] M. Shanahan. Robotics and the common sense informatic situation. In W. Wahlster, editor, *Proceedings of the ECAI*, pages 684–688, Budapest, Hungary, August 1996. John Wiley.

[Shoham, 1988] Y. Shoham. Chronological ignorance: Experiments in nonmonotonic temporal reasoning. *Artificial Intelligence*, 36:279–331, 1988.

[Stein and Morgenstern, 1994] L. A. Stein and L. Morgenstern. Motivated action theory: A formal theory of causal reasoning. *Artificial Intelligence*, 71:1–42, 1994.

[Thielscher, 1995] M. Thielscher. The logic of dynamic systems. In C. S. Mellish, editor, *Proceedings of the IJCAI*, pages 1956–1962, Montreal, Canada, August 1995. Morgan Kaufmann.

[Thielscher, 1996] M. Thielscher. Causality and the qualification problem. In L. C. Aiello, J. Doyle, and S. Shapiro, editors, *Proceedings of the KR*, pages 51–62, Cambridge, MA, November 1996. Morgan Kaufmann.

[Thielscher, 1997] M. Thielscher. Ramification and causality. *Artificial Intelligence*, 89(1–2):317–364, 1997.

[Thielscher, 1998a] M. Thielscher. Event Minimization with the Fluent Calculus. Technical Report TUD/FI98-03, Department of Computer Science, Dresden University of Technology, March 1998. Electronically available at `www.intellektik.informatik.tu-darmstadt.de/~mit`.

[Thielscher, 1998b] M. Thielscher. From Situation Calculus to Fluent Calculus: A New Motivation. Manuscript, 1998. Electronically available at `www.intellektik.informatik.tu-darmstadt.de/~mit`.

[Thielscher, 1998c] M. Thielscher. Steady versus stabilizing state constraints. In *Proceedings of the Symposium on Logical Formalizations of Commonsense Reasoning*, pages 239–248, London, UK, January 1998.

Logic Programming–based Representations I

A Comparison of the Static and the Disjunctive Well-founded Semantics and its Implementation

Stefan Brass
Computer Science
University of Hildesheim
Marienburger Platz 22
31141 Hildesheim, Germany
brass@uni-hildesheim.de

Jürgen Dix
Computer Science
University of Koblenz
Rheinau 1
56075 Koblenz, Germany
dix@uni-koblenz.de

Ilkka Niemelä
Computer Science
Helsinki University of Tech.
P.O.Box 1100
02015 HUT, Finland
Ilkka.Niemela@hut.fi

Teodor C. Przymusinski*
Computer Science
University of California
Riverside
CA 92521, USA
teodor@cs.ucr.edu

Abstract

In recent years, much work was devoted to the study of theoretical foundations of *Disjunctive Logic Programs* and *Disjunctive Deductive Databases*. While the semantics of non-disjunctive programs is fairly well understood the declarative and computational foundations of disjunctive programming proved to be much more elusive and difficult. Recently, two new and very promising semantics have been proposed for the class of disjunctive logic programs. Both of them extend the wellfounded semantics of normal programs. The first one is the *static semantics* proposed by Przymusinski and the other is the *disjunctive wellfounded semantics* proposed by Brass and Dix. Although the two semantics are based on very different ideas, we show in this paper that they turn out to be very closely related. In fact, we show that it is possible to restrict the underlying language of STATIC to get D-WFS. We also show how to use this characterization for an implementation based a circumscriptive theorem prover.

1 Introduction

Recently, considerable interest and research effort has been devoted to disjunctive logic programming, disjunctive deductive databases and to various extensions of non-monotonic formalisms ensuring a proper treatment of disjunctive information.

There are good reasons justifying this extensive research effort. In natural discourse as well as in various programming applications we often use *disjunctive statements*. One particular example of such a situation is *reasoning by cases*. Other obvious examples include:

- *Approximate information:* for instance, an age "around 30" can be 28, 29, 30, 31, or 32;
- *Legal rules:* the judge always has some freedom for his decision, otherwise he/she would not be needed; so laws cannot have unique models;
- *Diagnosis:* only at the end of a fault diagnosis we may know exactly which part of some machine was faulty but as long as we are searching, different possibilities exist;
- *Biological inheritance:* if the parents have blood groups A and 0, the child must also have one of these two blood groups (example from [Lip79]);
- *Natural language understanding:* here there are many possibilities for ambiguity and they are represented most naturally by multiple intended models;
- *Reasoning about concurrent processes:* since we do not know the exact sequence in which certain operations are performed, again multiple models come into play;
- *Conflicts in multiple inheritance:* if we want to keep as much information as possible, we should assume disjunction of the inherited values [BL93].

Formalisms promoting disjunctive reasoning are more *expressive* and *natural to use* since they permit direct translation of disjunctive statements from natural language and from informal specifications. The additional expressive power of disjunctive logic programs [EG93, EGM94, EG96] significantly simplifies the problem of *translation* of non-monotonic formalisms into logic programs, and, consequently, facilitates using logic programming as an *inference engine*

*Partially supported by the National Science Foundation grant #IRI-9313061.

for non-monotonic reasoning. Moreover, extensive recent work devoted to theoretic and algorithmic foundations of disjunctive programming suggests that there are good prospects for *extending the logic programming paradigm* to disjunctive programs.

However, the issue of finding a suitable semantics for disjunctive programs and databases proved to be far more complex than it was in the case of normal, non-disjunctive programs[1]. Quite recently, however, two new and very promising semantics have been proposed for the class of disjunctive logic programs. Both of them extend the wellfounded semantics of normal programs. The first one is the *static semantics* proposed by Przymusinski in [Prz95] and the other is the *D-WFS semantics* proposed by Brass and Dix in [BD94]. The two semantics are based on very different intuitions and are constructed by completely different means. While *D-WFS* is defined as the least semantics that is invariant under certain natural program transformations, the static semantics is obtained by first translating a logic program into a belief theory in the *Autoepistemic Logic of Beliefs (AEB)* and then constructing its *least static expansion* (defined as the least fixed point of a natural monotonic operator).

This paper is devoted to an in-depth study of the relationship between the two semantics. In spite of the fact that the two semantics attempt to provide a meaning for arbitrary disjunctive programs based on very different ideas, they turn out to be closely related. It is fairly easy to show that the static semantics satisfies the abstract properties on which the definition of D-WFS is based and therefore it is always at least as strong as D-WFS. However, because of the fact that the two semantics are defined using quite *different languages*, with a much richer language allowed by the static semantics, it is easy to give examples showing that, in general, the two semantics are different.

However, it turns out that when they are restricted to a common language, which we call *data base logic*, the two semantics become entirely *equivalent*. This fundamental result uses a nice and powerful characterization of STATIC$_{DBL}$, the static semantics restricted to the language $\mathcal{L}_{DBL}$. We consider this result to be quite significant. The *D-WFS* semantics [BD94] is based on the idea of generating the smallest "well-behaved" disjunctive semantics, i.e., the smallest semantics which satisfies some intuitive and natural structural properties. On the other hand, the static semantics [Prz95] is based on viewing disjunctive logic programs as special epistemic theories in which (introspective) beliefs are based on minimal model entailment (circumscription). The original papers [BD94, Prz95] (see also [Prz98, BD98a, BD98b]) contain a thorough discussion of the two semantics, compare them to other semantics and argue that they both represent natural and intuitive semantics for disjunctive programs. Our main result shows that as long as both semantics are restricted to the common language they become identical and therefore combine the benefits of both semantics. Consequently, this discovery constitutes yet one more powerful argument in favor of the two semantics.

Our main result not only clarifies the declarative meaning of the two semantics. It turns out that it also allows us to further clarify their *procedural semantics* by establishing new computational results. Namely, recently one of the authors [Nie96b, Nie96a] introduced a novel method for implementing circumscription (with fixed and varying predicates) which has polynomial space complexity and is able to handle circumscriptive reasoning also in the case when the set of premises has a large number of minimal models. Our second main result shows how this method can be nicely adapted for the computation of the D-WFS and static semantics.

Our results are obtained by using the characterization of least static expansions from [BDP96] and a characterization of D-WFS contained in [BD98a].

The paper is organized as follows. In the next Section 2 we recall the definition and basic properties of disjunctive logic programs and semantics as well as the modal framework of the *autoepistemic logic of beliefs* (AEB). In Section 3 we briefly recall the semantics D-WFS and STATIC. We introduce in Section 4 the restricted language $\mathcal{L}_{DBL}$ with non-nested beliefs, *database logic*, and characterize the least static expansion in this language — this defines the semantics STATIC$_{DBL}$. We then show, using our characterization, that STATIC$_{DBL}$ coincides with D-WFS: STATIC$_{DBL}(P)$ = D-WFS(P). In Section 5 we consider how to use our characterizations for an efficient implementation based on methods introduced recently by Niemelä in [Nie96b, Nie96a]. Related approaches are discussed in Section 6. We end with the conclusions in Section 7.

2 Preliminaries

We introduce in this section the necessary notions used throughout the paper. In Subsection 2.1 we consider disjunctive programs and semantics, while in Subsection 2.2 we introduce the modal framework of the au-

[1]The book by Minker et. al. [LMR92] provides a detailed and well-organized account of the extensive research effort in this area. See also [Dix95, Min96].

toepistemic logic of beliefs.

2.1 Disjunctive Logic Programs and Their Semantics

We consider instantiated programs over some fixed finite language $\mathcal{L}$ containing the binary connectives $\vee, \wedge, \leftarrow$, the unary connective *not* and the *falsum* $\bot$. We write $At_{\mathcal{L}}$ for the set of all atoms different from $\bot$.

Definition 2.1 (Logic Program P, heads(P))
A logic program P is a finite set of rules of the form

$$A_1 \vee \cdots \vee A_k \leftarrow B_1 \wedge \cdots \wedge B_m \wedge not\, C_1 \wedge \cdots \wedge not\, C_n,$$

where the $A_i/B_i/C_i$ are $\mathcal{L}$-atoms different from $\bot$, $k \geq 1$, $m \geq 0$, $n \geq 0$. We allow empty conjunctions: they are considered as abbreviations for not $\bot$ (the verum*).*

We identify such a rule with the pair consisting of the following set of atoms $\mathcal{A} := \{A_1, \ldots, A_k\}$, and the set of atoms and not-atoms $\mathcal{B}ody := \{B_1, \ldots, B_m, not\, C_1, \ldots, not\, C_n\}$, and write it as

$$\mathcal{A} \leftarrow \mathcal{B} \wedge not\, \mathcal{C}.$$

We write **heads**(P) *for the set of all atoms occuring in rule heads in P.*

By pure *disjunctions we mean disjunctions consisting solely of positive or of negative literals.*

The set **heads**(P) captures with a simple syntactical condition atoms that are *possibly true.* This condition is used later to derive literals *not* A, namely if A is *not* contained in **heads**(P'), where P' is a transformed program obtained from P.

The general problem now is to define a *semantics* for the class of all disjunctive logic programs. A semantics usually is given by a set of intended models. An even more general notion is given by

Definition 2.2 (Operator $\mid\!\sim$, Semantics $\mathcal{S}_{\mid\sim}$)
By a semantic operator $\mid\!\sim$ we mean a binary relation between logic programs and pure disjunctions (i. e. disjunctions of the form $A_1 \vee \ldots \vee A_n$ or $\neg A_1 \vee \ldots \vee \neg A_n$) which satisfies the following three arguably obvious conditions:

1. *If $P \mid\!\sim \psi$ and $\psi \subseteq \psi'$, then $P \mid\!\sim \psi'$* (Right Weakening).
2. *If $\mathcal{A} \leftarrow true \in P$ for a disjunction $\mathcal{A}$, then $P \mid\!\sim \mathcal{A}$* (Necessarily True).
3. *If $A \notin$* **heads**(P) *for any $\mathcal{L}$-atom A, then $P \mid\!\sim \neg A$* (Necessarily False).

Given such an operator $\mid\!\sim$ and a logic program P, by the semantics $\mathcal{S}_{\mid\sim}(P)$ of P determined by $\mid\!\sim$ we mean the set of all pure disjunctions derivable by $\mid\!\sim$ from P, i.e., $\mathcal{S}_{\mid\sim}(P) := \{\psi \mid P \mid\!\sim \psi\}$.

It will become clear later why we represent negative disjunctions by $\neg A_1 \vee \ldots \vee \neg A_n$ and not by *not* $A_1 \vee \ldots \vee$ *not* A_n. Note that any of the known semantics in logic programming induces such a formal notion of semantics.

2.2 The Logic of Autoepistemic Beliefs

The language of AEB is a propositional modal language $\mathcal{L}_{AEB}$ obtained by augmenting the classical propositional language $\mathcal{L}$ with a modal operator $\mathbf{B}$, called the *belief* operator. The atomic formulae of the form $\mathbf{B}F$, where F is an arbitrary formula of $\mathcal{L}_{AEB}$, are called *belief atoms.* The formulae of $\mathcal{L}$ are called *objective.* Any theory T in the language $\mathcal{L}_{AEB}$ is called a *belief theory.* Observe that arbitrarily deep levels of *nested beliefs* are allowed in belief theories.

We assume the following two simple axiom schemata and one inference rule describing the arguably obvious properties of belief atoms:

(D) Consistency: $\neg\mathbf{B}\bot$

(K) Normality: $\mathbf{B}(F \rightarrow G) \rightarrow (\mathbf{B}F \rightarrow \mathbf{B}G)$
for any formulae F,G

(N) Necessitation Rule: $\frac{F}{\mathbf{B}F}$ for any formula F.

The first axiom states that tautologically false formulae are *not* believed. The second axiom states that if we believe that a formula F implies a formula G and if we believe that F is true then we believe that G is true as well. The necessitation inference rule states that if a formula F has been proven to be true then F is believed to be true.

Definition 2.3 (Derivability from $\mathcal{L}_{AEB}$)
For any belief theory T, we denote by $Cn_{AEB}(T)$ the smallest set of formulae of the language $\mathcal{L}_{AEB}$ which contains the theory T, all the (substitution instances of) the axioms (K) and (D) and is closed under both standard propositional consequence and the necessitation rule (N). We say that a formula F is derivable *from theory T in the logic AEB if F belongs to $Cn_{AEB}(T)$. T is* consistent *if the theory $Cn_{AEB}(T)$ is consistent.*

In the presence of axiom (K), the axiom (D) is equivalent to the axiom

(D′) Restricted Consistency: $\mathbf{B}F \rightarrow \neg\mathbf{B}\neg F$ for any formula F.

stating that if we believe in a formula F then we do *not* believe in $\neg F$. Moreover, it is easy to see that the axioms imply that the belief operator $\mathbf{B}$ obeys the *distributive law for conjunctions* (see [Prz95]).

(C) Distributivity wrt Conjunctions: For any formulae F and G:

$$\mathbf{B}(F \wedge G) \;\equiv\; \mathbf{B}F \wedge \mathbf{B}G \qquad (1)$$

The most important notion in the sequel is that of a *minimal* model:

Definition 2.4 (Min. Models in $\mathcal{L}_{AEB}$, [Prz95]) *By a* minimal model *of a belief theory T we mean a model M of T with the property that there is* no *smaller model N of T which coincides with M on belief atoms $\mathbf{B}F$. If a formula F is true in all minimal models of T then we write: $T \models_{\min} F$ and say that F is* minimally entailed[2] *by T.*

The intended meaning of belief atoms $\mathbf{B}F$ is based on *predicate minimization*:

$$\begin{aligned}\mathbf{B}F \quad &\equiv \quad F \text{ is minimally entailed}\\ &\equiv \quad F \text{ is true in all minimal models.}\end{aligned}$$

Accordingly, beliefs in AEB can be called *minimal beliefs* (see [Min82, GPP88] and [McC80]). As in Moore's Autoepistemic Logic, also in the Autoepistemic Logic of Beliefs the intended meaning of belief atoms is implemented by defining plausible sets of beliefs that an ideally rational and introspective agent may hold, given a set of premises T.

Definition 2.5 (Static AE Expansion, [Prz95]) *A belief theory T° is called a* static autoepistemic expansion *of a belief theory T if it satisfies the following fixed-point equation:*

$$T^\circ = Cn_{AEB}(T \;\cup \{\mathbf{B}F : T^\circ \models_{\min} F\}),$$

where F ranges over all formulae of $\mathcal{L}_{AEB}$.

It turns out that every belief theory T in AEB has the *least* (in the sense of set-theoretic inclusion) static expansion T° which has an *iterative* definition as the *least fixed point* of the monotonic[3] belief closure operator:

$$\Psi_T(S) = Cn_{AEB}(T \cup \{\mathbf{B}F : S \;\models_{\min} F\}),$$

where S is an arbitrary belief theory and the F's range over all formulae of $\mathcal{L}_{AEB}$.

[2]For readers familiar with *circumscription*, this means that we are considering predicate circumscription $CIRC(T;\mathcal{L})$ of the theory T in which atoms from the objective language are minimized while the belief atoms $\mathbf{B}F$ are fixed, i.e., $T \;\models_{\min} F \;\equiv\; CIRC(T;\mathcal{L}) \models F$. In other words, minimal models are obtained by first assigning *arbitrary* truth values to the belief atoms and then *minimizing* objective atoms. See also [YY93].

3 The semantics D-WFS and STATIC

In this section we review briefly the semantics D-WFS and STATIC to make the paper self-contained.

3.1 D-WFS

The semantics D-WFS was originally defined ([BD94, BD98b]) as the *weakest* semantics $\mid\!\sim$ which is invariant under certain natural transformations $Trans : P \mapsto Trans(P)$ mapping logic programs into logic programs.

Theorem 3.1 (D-WFS, [BD98b]) *There exists the weakest semantics $S_{\mid\sim}$ which satisfies the properties of Generalized Principle of Partial Evaluation (GPPE), Elimination of Tautologies (TAUT), Non-Minimal Rules (NMIN), Positive and Negative Reduction (RED$^+$, RED$^-$). Moreover, this semantics is consistent (i.e. it does not derive a literal A and its negation $\neg A$) and closed under logical consequences (as a set consisting of pure disjunctions). We call it the Disjunctive Wellfounded Semantics, or, briefly, D-WFS. It coincides with the wellfounded semantics WFS for normal and with GCWA for positive disjunctive programs.*

It remains to illustrate the properties mentioned.

TAUT: Delete a rule $\mathcal{A} \leftarrow \mathcal{B} \wedge not\,\mathcal{C}$ with $\mathcal{A} \cap \mathcal{B} \neq \emptyset$.

NMIN: Delete a rule $\mathcal{A} \leftarrow \mathcal{B} \wedge not\,\mathcal{C}$ if there is another rule $\mathcal{A}' \leftarrow \mathcal{B}' \wedge not\,\mathcal{C}'$ with $\mathcal{A}' \subseteq \mathcal{A}$, $\mathcal{B}' \subseteq \mathcal{B}$, and $\mathcal{C}' \subseteq \mathcal{C}$.

RED$^+$: Replace a rule $\mathcal{A} \leftarrow \mathcal{B} \wedge not\,\mathcal{C}$ by $\mathcal{A} \leftarrow \mathcal{B} \wedge not\,(\mathcal{C} \cap heads(\Phi))$.

RED$^-$: Delete a rule $\mathcal{A} \leftarrow \mathcal{B} \wedge not\,\mathcal{C}$, if there is a rule $(\mathcal{A}' \leftarrow)$ with $\mathcal{A}' \subseteq \mathcal{C}$.

GPPE: Replace a rule $\mathcal{A} \leftarrow \mathcal{B} \wedge not\,\mathcal{C}$ where $\mathcal{B}$ contains a distinguished atom B by the k rules

$$\mathcal{A} \cup (\mathcal{A}_i \setminus \{B\}) \;\leftarrow\; (\mathcal{B} \setminus \{B\}) \cup \mathcal{B}_i \;\wedge\; not\,(\mathcal{C} \cup \mathcal{C}_i)$$

[3]Strictly speaking the operator is only monotone on the lattice of all theories of the form $T \cup M_{bel}$ where T is fixed and M_{bel} ranges over all sets of belief formulae.

where $\mathcal{A}_i \leftarrow \mathcal{B}_i \wedge not\, \mathcal{C}_i$ $(i = 1, \ldots, k)$ are *all* the rules with $B \in \mathcal{A}_i$.

The last rule has been introduced for disjunctive programs independently by Sakama/Seki and Brass/Dix in [SS94, BD94]. It allows us to get rid of positive body atoms. Given a program rule, we are free to select a specific positive occurrence of an atom B and then resolve this rule with all other rules that contain B in their heads (this set of rules constitutes the definition of B). Thus we replace the original program rule with a set of new rules. GPPE also covers the degenerate case when the atom B to be replaced does not appear in any head: then the whole rule is simply deleted.

In fact, we discovered later ([BD98a]) that our set of transformations is also *confluent* and terminating. This is of special importance, because given a program P, we can just apply our transformations in an arbitrary order and eventually reach a normal form res(P). From this normal form res(P), we can immediately read off the semantics as follows

$$P \mid\!\sim_{\text{D-WFS}} \psi \text{ iff } \begin{cases} \exists \mathcal{A} \subseteq \psi \text{ with } (\mathcal{A} \leftarrow) \in \text{res}(P) \text{ or} \\ \exists \neg A \in \psi \text{ and } A \notin \textbf{heads}(\text{res}(P)). \end{cases}$$

This is also the best method to really compute D-WFS for small examples and we will use it in the sequel. However in order to link D-WFS with a theorem prover and also with STATIC, we need a completely different characterization. This is a very technical characterization and had been stated in [BD98a]. It is similar to the Gelfond/Lifschitz translation and uses the notion of *reducing a program P with respect to a set of disjunctions Dis*:

Definition 3.2 (P/Dis)
Let P be a disjunctive logic program and let Dis be a set of pure disjunctions. Let P/Dis be the program obtained from P by doing the following reductions for all $not\, C$ and $C_1 \vee \ldots \vee C_k$

- *if $\neg C \in Dis$, then remove all occurrences of $not\, C$,*
- *if $C_1 \vee \ldots \vee C_k \in Dis$ then remove all rules that contain $\{not\, C_1, \ldots, not\, C_k\}$ in their bodies.*

P/Dis is obviously a slight generalization of the Gelfond-Lifschitz transformation. While the latter is defined relative to a set $N \subseteq At_{\mathcal{L}}$ in such a way that P/N is always positive, *our P/Dis still is a disjunctive program containing possibly the not-atoms. In fact, the GL-transform can be obtained from our transform by setting $P/N = P/Dis_N$ where $Dis_N := N \cup \{\neg X : X \in At_{\mathcal{L}} \setminus N\}$.*

Definition 3.3 (SEM$_0$)
Let P be a disjunctive logic program over $\mathcal{L}$. We define a set $Dis(P)$ of pure disjunctions as follows

$Dis(P) := Dis^+(P) \cup Dis^-(P)$ where

$Dis^+(P)$ is the following set of disjunctions

$$\{ A_1 \vee \ldots \vee A_k : P/At_{\mathcal{L}} \models A_1 \vee \ldots \vee A_k \}$$

and $Dis^-(P)$ is the following set of negative literals

$$\{ \neg A : P/Dis_N \models_{\min} \neg A \text{ for all } N \subseteq At_{\mathcal{L}} \text{ s. t. } N \models Dis^+(P)\}$$

Here, the series D_i defined by $D_{n+1} := Dis(P/D_n)$ grows monotonically and eventually gets constant. We define $SEM_0(P)$ to be the limit of this series.

The underlying idea in the last definition of a semantics SEM$_0$ is to use $P/At_{\mathcal{L}}$ (i.e. we just delete all rules containing negative literals) for deriving positive disjunctions and to use P/Dis_N (i.e. we set all negative literals outside N to true) for deriving negative literals. $\models_{\min}$ stands for the usual notion of *"truth in all minimal models"* where *minimal* is used classically. Note that since P/Dis_N is just a positive disjunctive program, this notion coincides with that introduced in Definition 2.4 because no belief-atoms occur.

Let us briefly comment on the occurrence of the set N in the $Dis^-(P)$ Definition 3.3. Suppose we have the program "$A \vee B \leftarrow,\ C \leftarrow not\, A \wedge not\, B$". Obviously of all our transformations only RED$^-$ can be applied and leads to the program "$A \vee B \leftarrow$" which constitutes the normal form. Therefore $not\, C$ is derivable in D-WFS, because it does not appear in **heads**$(\{A \vee B \leftarrow\})$.

Now if we remove the proviso "$N \models Dis^+(P)$" in the definition of $Dis^-(P)$, then we also have to consider $N_0 := \emptyset$, which implies that $\neg C$ would not be included. This proviso therefore guarantees that only those $N \subseteq At_{\mathcal{L}}$ that are compatible with $Dis^+(P)$ are considered.

Theorem 3.4 (D-WFS = SEM$_0$, ([BD98a]))
D-WFS is complete with respect to SEM_0:

$$\text{D-WFS}(P) = SEM_0(P) \text{ for all } P.$$

3.2 STATIC

Here we show how the general framework introduced in Subsection 2.2 can be naturally used to define a semantics for disjunctive logic programs.

Of course, any disjunctive logic program in the sense of Definition 2.1 can be seen as a theory in $\mathcal{L}_{AEB}$ by replacing *not* simply by $\neg$. Unfortunately, this would not give the right interpretation of the $\leftarrow$-arrow used in logic programming (there, usually, contrapositions are not allowed) and it would also not correspond to the various meanings of ***negation-as-failure***.

But it turns out that a precise meaning of negation by default can be captured by appropriate translations of a disjunctive logic program into a belief theory. In our approach this meaning will be defined by

translating *not* F into the formula $\mathbf{B}\neg F$ in AEB

thus providing *not* F with the intended meaning of ***"$\neg F$ is minimally entailed"*** or ***"F is false in all minimal models"***.

Definition 3.5 (Belief-Programs)
A belief-program is a (finite or infinite) set of clauses of the form:

$A_1 \vee \ldots \vee A_k \leftarrow B_1 \wedge \ldots \wedge B_m \wedge \mathbf{B}\neg C_1 \wedge \ldots \wedge \mathbf{B}\neg C_n$,

where A_i's, B_i's, and C_i's are $\mathcal{L}$-atoms and $k > 0$.

Therefore any disjunctive logic program P in the sense of Definition 2.1 can be translated into a special belief theory in $\mathcal{L}_{AEB}$ – the associated belief program. The meaning of *not* is given by *not* F $\equiv$ F is false in all minimal models $\equiv$ $\neg F$ is minimally entailed.

Thus STATIC can be seen as a semantics for disjunctive logic programs in the sense of Definition 2.2:

Definition 3.6 (STATIC)
Given a logic program P, we translate it into a belief program, compute the least static expansion and consider all formulae of the form $A_1 \vee \ldots \vee A_n$ or $\mathbf{B}\neg A_1 \vee \ldots \vee \mathbf{B}\neg A_m$ true in it. Their re-translations $A_1 \vee \ldots \vee A_n$ resp. $\neg A_1 \vee \ldots \vee \neg A_m$ constitute the set of derivable pure disjunctions: we call this set STATIC(P).

While arbitrary belief theories do not necessarily have consistent least static expansions, belief programs do. In fact, they can be characterized nicely as follows ([BDP96]):

Theorem 3.7 (Least Static Expansion in $\mathcal{L}_{AEB}$)
Every belief program P in AEB has the least static *expansion, which is consistent. It is obtained as the least fixed point $P^\diamond$ of the following monotonic belief closure operator Ψ_P, which assigns to a set S the closure*

$$Cn(P \cup \{\mathbf{B}(\neg p_1) \wedge \cdots \wedge \mathbf{B}(\neg p_n) \rightarrow \mathbf{B}(\neg q_1 \vee \ldots \vee \neg q_m) : S \models_{min} \neg p_1 \wedge \cdots \wedge \neg p_n \rightarrow (\neg q_1 \vee \ldots \vee \neg q_m)\}),$$

The sequence $\{P^\alpha\}$, defined by $P^0 := \emptyset$, $P^{\alpha+1} := \Psi_P(P^\alpha)$, $P^\gamma := \bigcup_{\beta \leq \gamma} \Psi_P(P^\beta)$, is monotonically increasing and has a unique fixed point $P^\diamond = P^\lambda = \Psi_P(P^\lambda)$, for some ordinal λ. For finite theories P the fixed point $P^\diamond$ is reached after finitely many steps.

4 A Restriction of STATIC and its equivalence to D-WFS

We start by considering an example illustrating a setting where STATIC is stronger than D-WFS.

Example 4.1 [STATIC is stronger than D-WFS]
Let

$$\begin{array}{llll} P_1: & a \vee c & \leftarrow & not\, d \\ & c & \leftarrow & not\, c \\ & d & \leftarrow & c \end{array}$$

Now *STATIC* concludes $\neg a$ but D-WFS does not. Let us explain why.

The normal form res(P_1) is obtained by applying GPPE to the third clause (replacing c). Thus res(P_1) consists of the first two clauses and the following two clauses $d \leftarrow not\, c$ and $d \vee a \leftarrow not\, d$. Consequently, by our characterization of D-WFS in terms of the normal form, all atoms are undefined.

For *STATIC* we have to consider the corresponding belief theory. Due to the (K)-axiom we have $\mathbf{B}\neg c \leftarrow \mathbf{B}\neg d$ in that belief theory. Together with the second clause we have $c \leftarrow \mathbf{B}\neg d$. Therefore, if we look at the first clause, a has no chance to be true in minimal models: $\mathbf{B}\neg d$ does not hold (because then both $\mathbf{B}c$ and $\mathbf{B}\neg c$ would hold). Therefore $\mathbf{B}\neg d$ holds, but then c must also hold and a need not to hold.

The last example shows that the (K)-Axiom in STATIC is powerful enough to use contrapositives and thus it is stronger than D-WFS. The idea to restrict STATIC is to replace (K) by a weaker axiom and also to work in a language where there are no nested beliefs.

4.1 STATIC in a restricted language $\mathcal{L}_{DBL}$

As for $\mathcal{L}_{AEB}$ we start with an objective language $\mathcal{L}$ and augment it by belief atoms of the form $\mathbf{B}F$ where F is a ***positive*** or ***negative*** formula from $\mathcal{L}$: this gives us the language $\mathcal{L}_{DBL}$. Thus, $\mathcal{L}_{DBL}$-formulae do not contain nested beliefs.

Definition 4.2 (Formulae Derivable in $\mathcal{L}_{DBL}$)
For any belief theory T, we denote by $Cn_{DBL}(T)$ the smallest set of formulae of the language $\mathcal{L}_{DBL}$ which contains the theory T, all the (substitution instances of) the axioms (C) and (D') and is closed under both

standard propositional consequence and the necessitation rule (N). We say that a formula F is derivable *from theory T in the logic DBL if F belongs to* $Cn_{DBL}(T)$. *A belief theory T is* consistent *if the theory* $Cn_{DBL}(T)$ *is consistent.*

As before, we call a theory $T^\diamond$ *a static autoepistemic expansion, if it satisfies*

$$T^\diamond = Cn_{DBL}(T \cup \{\mathbf{B}F : T^\diamond \models_{\min} F\}).$$

Our next theorem shows that the least static expansion of a belief program in $\mathcal{L}_{DBL}$ is consistent and can be computed as follows:

Theorem 4.3 (Least Static Expansion in $\mathcal{L}_{DBL}$) *Every belief program P in DBL has the* least *static expansion, which is consistent. In addition, the set of pure disjunctions contained in this least expansion is already contained in the least fixed point* $P^\diamond$ *of the following monotonic belief closure operator* Ψ_P, *which assigns to a set S the closure*

$$\begin{aligned} Cn(P \cup \{&\neg\mathbf{B}\neg q_1 \vee \ldots \vee \neg\mathbf{B}\neg q_k : S \models q_1 \vee \ldots \vee q_k\} \\ \cup\{&\mathbf{B}(\neg p_1 \vee \ldots \vee \neg p_n) : S \models_{\min} \neg p_1 \vee \ldots \vee \neg p_n\}), \end{aligned}$$

where Cn denotes the consequences of propositional logic and the p_i, q_i *are objective atoms. The sequence* $\{P^\alpha\}$, *defined by* $P^0 := \emptyset$, $P^{\alpha+1} := \Psi_P(P^\alpha)$, $P^\gamma := \bigcup_{\beta \leq \gamma} \Psi_P(P^\beta)$, *is monotonically increasing and has a unique fixed point* $P^\diamond = P^\lambda = \Psi_P(P^\lambda)$, *for some ordinal* λ. *For finite theories P the fixed point* $P^\diamond$ *is reached after finitely many steps.*

Proof. (Sketch): It is easy to show that the operator is monotonic and thus has a least fixed-point $P^\diamond$. However, it is also immediate that this fixpoint is not the least static expansion, simply because not all necessary belief atoms are contained. Note that in the construction only the most simplified belief atoms were added: instead of adding $\mathbf{B}(q_1 \vee \ldots \vee q_k)$ the disjunction $\neg\mathbf{B}\neg q_1 \vee \ldots \vee \neg\mathbf{B}\neg q_k$ was added thereby applying axiom (C), (D′) and (N). It is important to note that the belief program P used in the construction just contains belief atoms of the most simple form: $\mathbf{B}(\neg q_1)$. This implies that the real least static expansion is obtained by "closing" $P^\diamond$ under the axioms (C), (D′) and (N). This only introduces new beliefs of the form $\mathbf{B}(\phi)$ where ϕ is a (positive or negative) formula such that $\mathbf{B}(\phi')$ (where ϕ' is a *subformula* of ϕ) is already present. Thus it has no effect on the set of pure disjunctions. □

The last theorem gives rise to a semantics STATIC_{DBL} in the same way as STATIC was introduced in Definition 3.6. This means that $\text{STATIC}_{DBL}(\text{P})$ is a certain set of pure disjunctions. In fact we have the following corollary which will be used later to establish the equivalence of D-WFS and STATIC_{DBL}:

Corollary 4.4 (STATIC_{DBL})
The set of pure disjunctions contained in the least static expansion in $\mathcal{L}_{DBL}$ *is already obtained as the least fixpoint of the following simplified closure operator* $\Psi'_P(S)$

$$\begin{aligned} Cn(P \cup \{&\neg\mathbf{B}\neg q_1 \vee \ldots \vee \neg\mathbf{B}\neg q_k : S \models q_1 \vee \ldots \vee q_k\} \\ \cup\{&\mathbf{B}(\neg p_1) : S \models_{\min} \neg p_1\}), \end{aligned}$$

Corollary 4.4 follows from Theorem 4.3 because the corresponding belief theory does not contain any axioms that relate belief atoms $\mathbf{B}(p \vee q)$ with $\mathbf{B}(q)$ and $\mathbf{B}(p)$! If this were the case, then the last characterization (which only uses belief atoms of the form $\mathbf{B}(\phi)$ where ϕ is a negated atom) would certainly be different from the one in Theorem 4.3, because there much more belief atoms (those where ϕ is a disjunction of negated atoms) are contained. Therefore the choice of the underlying axioms for $\mathcal{L}_{DBL}$ is very important.

4.2 STATIC_{DBL} coincides with D-WFS

We aim to reformulate the characterization of D-WFS given in Theorem 3.4 in order to bring it closer to minimal entailment $\models_{\min}$ of Definition 2.4 which is used in Lemma 4.4.

As we are considering disjunctive logic programs where we do not have belief atoms but *not* (ϕ) atoms, minimal entailment means that *not* (ϕ) atoms are fixed and all objective atoms are minimized. This is just to facilitate notation: fixing *not* (ϕ) atoms is completely equivalent to fixing $\mathbf{B}(\neg\phi)$ atoms.

To capture the notion of minimal entailment used in Definition 3.3 we strengthen the notion of models for a disjunctive logic program P by requiring that a model I (a set of atoms and formulae *not* (ϕ)) of P has to be *NAF-consistent* for P, i.e., it must satisfy the following condition:

$$\text{If } P \models a_1 \vee \cdots \vee a_n, \text{ then } \exists i \in \{1, \ldots, n\}, \mathit{not}\,(a_i) \notin I.$$

For example, $\{A, C, \mathit{not}\,A, \mathit{not}\,B\}$ is not a NAF-consistent model of $P = \{A \vee B \leftarrow, C \leftarrow \mathit{not}\,A \wedge \mathit{not}\,B\}$ but $\{A, C, \mathit{not}\,A\}$ is. However, the latter is not a minimal model of P.

If the operator *Dis* employed in the characterization of Theorem 3.4 is modified in the following way, D-WFS is captured using normal minimal entailment but with respect to NAF-consistent models. We denote this entailment relation by $\models_{\min(N)}$.

$$\begin{array}{rl} Dis_{\min}(P) = & \{A_1 \vee \cdots \vee A_k | P \models A_1 \vee \cdots \vee A_k\} \cup \\ & \{\neg A \quad\quad |P \models_{\min(N)} \neg A\} \end{array}$$

We denote by $\text{SEM}_c(P)$ the semantics which is obtained by iterating this operator and propagating the results as in SEM_0 in Theorem 3.4. This gives us exactly D-WFS.

Theorem 4.5 *$SEM_0(P) = \text{SEM}_c(P)$.*

The theorem can be proved by establishing the following two propositions.

Proposition 4.6 *For any set of atoms $\{A_1, \cdots, A_k\}$,*

$$P/At_{\mathcal{L}} \models A_1 \vee \cdots \vee A_k \textit{ iff } P \models A_1 \vee \cdots \vee A_k.$$

Proposition 4.7 *For any disjunctive logic program P and propositional formula ϕ,*

$$P/Dis_N \models_{\min} \phi \textit{ for all } N \subseteq At_{\mathcal{L}} \textit{ with } N \models Dis^+(P)$$
$$\textit{iff}$$
$$P \models_{\min(N)} \phi.$$

Theorem 4.8 (STATIC_{DBL} = D-WFS) *$STATIC_{DBL}$ coincides with D-WFS.*

Proof. (Sketch): We have to compare the construction of D-WFS based on $\models_{\min(N)}$ and the construction of STATIC_{DBL} based on Corollary 4.4.

While in the first construction every step is split into deriving certain disjunctions and doing certain program transformations (to prepare the next step), the latter is given by applying an operator and getting larger and larger theories.

It is easy to see that the program transformations of the first construction are simulated in the latter method by the introduction of the belief atoms: new belief atoms restrict in the next round the set of minimal models used to derive negative disjunctions. To be more precise, the first transformation in Definition 3.2 is captured by adding $\mathbf{B}(\neg C)$ in the definition of the operator Ψ' in Corollary 4.4. The second transformation in Definition 3.2 is simulated by adding $\neg\mathbf{B}\neg C_1 \vee \ldots \vee \neg\mathbf{B}\neg C_k$. NAF-consistent models are nothing else than arbitrary minimal models (as used in the construction of Corollary 4.4) satisfying the additional constraints formalized by the belief atoms.

In fact, in the limit we get the same results. □

5 Towards an Implementation

The results established in the previous sections show interesting connections between minimal entailment and the D-WFS and STATIC semantics. They imply that D-WFS (and thus STATIC_{DBL}) can be implemented using iterative minimal model reasoning with a fairly standard notion of minimal entailment $\models_{\min(N)}$ (which is standard circumscription but over NAF-consistent models). In this section we demonstrate how this can be done by exploiting directly the new characterization of D-WFS in Theorem 4.5. Then we discuss how the required notion of minimal entailment $\models_{\min(N)}$ can be implemented using a new tableau method developed for circumscription [Nie96b, Nie96a, BFN96]. An interesting feature of the method is that it enables an implementation technique for D-WFS which works in *polynomial space*. This is important when dealing with larger programs.

The idea is that D-WFS is implemented as an iterative reduction on disjunctive logic programs. The reduction starts with the original program P and leads to a *reduced* program P^* with the property that every query can be answered from this program with one theorem prover call. This can be seen as a compilation (or a partial evaluation) of the program leading to a smaller program from which all queries can be answered.

The reduced program is obtained as a "fixed point" of a reduction operator $\text{R}_\text{D}(\cdot)$ which is derived from the operator used in SEM_c.

$$\begin{array}{rcl} \text{R}_\text{D}(P) & = & \{\mathcal{A} \leftarrow \mathcal{B} \wedge not\, \mathcal{C}' \mid \mathcal{A} \leftarrow \mathcal{B} \wedge not\, \mathcal{C} \in P, \\ & & P/At_{\mathcal{L}} \not\models C_1 \vee \cdots \vee C_n \text{ where} \\ & & \mathcal{C} = \{C_1, \ldots, C_n\}, \\ & & \mathcal{C}' = \{C \in \mathcal{C} \mid P \not\models_{\min(N)} \neg C\}\} \end{array}$$

Now let P^* be the limit of the (monotonically decreasing) series of programs $P_0, P_1, \ldots$ where $P_0 = P$ and $P_{i+1} = \text{R}_\text{D}(P_i)$. Hence, the reduced program P^* is obtained by

- removing each rule such that $not\, C_1, \ldots, not\, C_n$ are all the negative body literals and $C_1 \vee \cdots \vee C_n$ is classically entailed by those rules of the remaining program with no negative body literals and
- removing all $not\,(C)$ for which $\neg C$ is minimally entailed by the remaining program

until no reduction is possible. It can be established that $P_i = P/D_i$ for all $i = 0, 1, \ldots$ Then we have the following theorem.

Theorem 5.1 $\mathrm{SEM}_c(P) = Dis_{\min}(P^*)$.

The theorem combined with Proposition 4.6 implies that a positive query can be answered using a classical theorem prover and those rules of the reduced program not containing negative body literals, i.e., $A_1 \vee \cdots \vee A_k \in$ D-WFS(P) iff $P^*/At_{\mathcal{L}} \models A_1 \vee \cdots \vee A_k$ but for negative queries minimal model reasoning is needed, i.e., $\neg A \in$ D-WFS(P) iff $P^* \models_{\min(\mathrm{N})} \neg A$.

The series of reductions is straightforward to implement[4] if decision procedures for classical entailment and minimal entailment $\models_{\min(\mathrm{N})}$ are available.

decision procedure for classical entailment and minimal entailment $\models_{\min(\mathrm{N})}$ is available. The latter procedure can be implemented using a new tableau method for circumscription presented in [Nie96b, Nie96a]. Next we explain briefly the main ideas in the method.

Standard tableau methods deciding whether a query follows (classically) from a set of premises can be seen as systematic attempts to construct a counter-model, i.e., a model where the premises are true and the query false. Each open branch in the tableau presents a potential counter-model and a tableau proof is found if all branches can be closed indicating that there are no counter-models. In principle, it seems that tableau methods can be easily extended to handle minimal model reasoning by accepting as counter-models only minimal models of the premises. Technically this is more complicated because it is not straightforward to decide whether a model provided by an open branch is a minimal model of the premises without actually considering exponentially many other models. We illustrate the difficulties with the program P

$$\begin{array}{c} a_1 \vee b_1 \leftarrow, \ldots, a_n \vee b_n \leftarrow, \\ a_1 \leftarrow a_0, \ldots, a_n \leftarrow a_{n-1} \end{array} \tag{2}$$

and the task of deciding whether $\neg a_0$ is minimally entailed by the program. A tableau method like that in [Nie96b, Nie96a] starts with the negation of the query and generates one counter-model $M = \{a_0, \ldots, a_n\}$. The problem is to decide whether this is a minimal model of P. A brute-force method would examine all $2^{n+1} - 1$ models that are smaller than M. In [Nie96b, Nie96a] an alternative approach is proposed by devising a novel characterization of minimal models which makes it possible to test whether a model is a minimal one independently of other models using one theorem prover call. The method can handle standard circumscription with fixed and varying predicates and for the notion of minimality in disjunctive programs it leads to the following characterization of minimal models. A model (a set of atoms and *not* (ϕ) atoms) of a program P is a minimal one iff

$$P \cup \mathrm{N}(M) \models A_1 \wedge \cdots \wedge A_n \tag{3}$$

where $A_1, \ldots, A_n$ are the (ordinary) atoms in M and

$\mathrm{N}(M) = \{\neg A \mid A$ is an (ordinary) atom, $A \notin M\} \cup \{not\, C \in$

For example, given the program P (2) and the model $M = \{a_0, \ldots, a_n\}$, $\mathrm{N}(M) = \{\neg b_1, \ldots, \neg b_n\}$ and the minimality of M can be determined by verifying whether $P \cup \{\neg b_1, \ldots, \neg b_n\} \models a_0 \wedge \cdots \wedge a_n$ holds. It is a trivial task for an efficient classical theorem prover to decide that this is not the case implying that M is not a minimal model of P and that $\neg a_0$ is minimally entailed by P.

In order to extend the method from standard minimal model entailment to $\models_{\min(\mathrm{N})}$ the only change is to require that only NAF-consistent models are accepted. The test for NAF-consistency can be implemented using the following observation: a model is NAF-consistent for P iff

$$P \not\models C_1 \vee \cdots \vee C_m \tag{4}$$

where $not\, C_1, \ldots, not\, C_m$ are all the *not* (ϕ) atoms in the model. Hence, a model of P is a NAF-consistent minimal model of P iff the conditions (3) and (4) hold for the model.

Example 5.2 Consider the program P

$$\begin{array}{lll} a \vee b & \leftarrow & \\ c & \leftarrow & not\, a \\ d & \leftarrow & not\, b \\ e & \leftarrow & c \wedge d \\ f & \leftarrow & not\, e \\ f & \leftarrow & not\, a \wedge not\, b \wedge not\, c \end{array}$$

and models $M_1 = \{a, not\, b, c, d, e\}$ and $M_2 = \{a, not\, a, not\, b, c, d, e\}$. For M_1 the condition (3) fails as $\mathrm{N}(M_1) = \{\neg b, \neg f, not\, b\}$ and, e.g., $P \cup \mathrm{N}(M_1) \not\models c$. For M_2, $\mathrm{N}(M_2) = \{\neg b, \neg f, not\, a, not\, b\}$ and the condition (3) holds but (4) does not as $P \models a \vee b$. In fact, there are no NAF-consistent minimal models of P where e is true and, hence, $P \models_{\min(\mathrm{N})} \neg e$. Thus when deriving the reduced program P^*, $not\, e$ is removed. Similarly, the last rule is removed since $a \vee b \vee c$ is classically entailed by the rules without negative literals.

As mentioned in the beginning of the section, the technique outlined above enables a ***polynomial space*** implementation of D-WFS . This is because both classical entailment and minimal model entailment $\models_{\min(\mathrm{N})}$ can

[4]See `<URL:http://www.uni-koblenz.de/ag-ki/DLP/systems/dwfs-mm.html>`.

be implemented in polynomial space. For the latter a polynomial space bound can be realized by developing the tableau one branch at a time and performing the minimality and NAF-consistency tests using a polynomial space classical theorem prover. When the two entailment relations can be decided in polynomial space, similar space bounds clearly hold for computing the reduced program and for query-evaluation, as well.

6 Related Work

Meanwhile some systems for disjunctive logic programming semantics exist or are in development. Besides systems for computing the classical stable semantics ([GL88, GL90]), D. Seipel ([Sei94]) implemented a system (`<URL:http://www-info1.informatik.uni-wuerzburg.de/database/DisLog/introduction.html>`) for computing various variants of the disjunctive semantics contained in [LMR92].

A group in Vienna (T. Eiter, G. Gottlob, N. Leone) is currently building a system called `dlv` [ELM+97]. This is a knowledge representation system, based on disjunctive logic programming, which offers front-ends to several advanced KR formalisms and implements the disjunctive stable semantics. The system runs in *polynomial space* and *single exponential time*, and is able to efficiently recognize and process syntactical subclasses of disjunctive logic programs which have lower computational complexity than the general case (like, e.g., programs with head-cycle free disjunction or stratified negation).

Implementation of D-WFS and its extensions is one of the main goals of the *DisLoP project*, undertaken by the Artificial Intelligence Research Group at the University of Koblenz (J. Dix and U. Furbach) ([ADN97a, ADN97b], see also `<http://www.uni-koblenz.de/ag-ki/DLP/>`). In contrast to the other approaches just mentioned, it aims at extending certain theorem-proving calculi (*restart model elimination* and *hyper tableau calculi*) for disjunctive logic programming.

Finally, we refer to [DFN97], which contains a collection of nonmonotonic system descriptions (including some handling disjunction) and to the following webpage which is actively maintained and contains information on various logic programming systems that concentrate on non-monotonic aspects (different kinds of negation, disjunction, abduction etc.): `<URL:http://www.uni-koblenz.de/ag-ki/LP/>`.

7 Conclusion

While the semantics of non-disjunctive programs is fairly well understood and there exist several implementations of the best known semantics (wellfounded semantics and stable semantics), the declarative and computational foundations of disjunctive programming proved to be much more elusive and difficult. However, the recent introduction of the two new, promising semantics for the class of disjunctive logic programs, namely, the *static semantics* proposed by Przymusinski [Prz95] and the *D-WFS semantics* proposed by Brass and Dix [BD94, BD98b], seems to significantly enhance our understanding of the theoretical and algorithmic aspects of disjunctive logic programs.

Both semantics extend the wellfounded semantics of normal programs and have very natural and regular features analogous to those shared by the wellfounded semantics. Although the two semantics are based on very different ideas, they turn out to be closely related as we have just shown. We note that also the disjunctive stable semantics can be captured in our framework (this has been shown in [BD97] for a specific set of transformation rules).

In this paper we conducted a detailed study of the relationship between the static and the D-WFS semantics in the class of *disjunctive logic programs*. We proved an equivalence result showing how to restrict STATIC to get D-WFS.

Subsequently, we demonstrated the applicability and importance of the above theorems by using them to devise an implementation technique building on a novel method for circumscriptive reasoning. An implementation based on this technique has been developed in the DisLoP-project[5] at the university of Koblenz.

Acknowledgments

We thank Alexander Bochman and several referees for useful comments on parts of the paper.

References

[ADN97a] Chandrabose Aravindan, Jürgen Dix, and Ilkka Niemelä. Dislop: A research project on disjunctive logic programming. *AI Communications*, 10:151–165, 1997.

[ADN97b] Chandrabose Aravindan, Jürgen Dix, and Ilkka Niemelä. DisLoP: Towards a Disjunctive Logic Programming System. In J. Dix,

[5]See `<URL:http://www.uni-koblenz.de/ag-ki/DLP/>`.

U. Furbach, and A. Nerode, editors, *Logic Programming and Non-Monotonic Reasoning, Proceedings of the Fourth International Conference*, LNAI 1265, pages 342–353, Berlin, June 1997. Springer.

[BD94] Stefan Brass and Jürgen Dix. A disjunctive semantics based on unfolding and bottom-up evaluation. In Bernd Wolfinger, editor, *Innovationen bei Rechen- und Kommunikationssystemen,* (IFIP '94-Congress, Workshop FG2: Disjunctive Logic Programming and Disjunctive Databases), pages 83–91, Berlin, 1994. Springer.

[BD97] Stefan Brass and Jürgen Dix. Characterizations of the Disjunctive Stable Semantics by Partial Evaluation. *Journal of Logic Programming*, 32(3):207–228, 1997. (Extended abstract appeared in: Characterizations of the Stable Semantics by Partial Evaluation *LPNMR, Proceedings of the Third International Conference, Kentucky*, pages 85–98, 1995. LNCS 928, Springer.).

[BD98a] Stefan Brass and Jürgen Dix. Characterizations of the Disjunctive Well-founded Semantics: Confluent Calculi and Iterated GCWA. *Journal of Automated Reasoning*, 20(1):143–165, 1998. (Extended abstract appeared in: Characterizing D-WFS: Confluence and Iterated GCWA. *Logics in Artificial Intelligence, JELIA '96*, pages 268–283, 1996. Springer, LNCS 1126.).

[BD98b] Stefan Brass and Jürgen Dix. Semantics of Disjunctive Logic Programs Based on Partial Evaluation. *Journal of Logic Programming*, accepted for publication, 1998. (Extended abstract appeared in: Disjunctive Semantics Based upon Partial and Bottom-Up Evaluation, *Proceedings of the 12-th International Logic Programming Conference, Tokyo*, pages 199–213, 1995. MIT Press.).

[BDP96] Stefan Brass, Jürgen Dix, and Teodor. C. Przymusinski. Super Logic Programs. In L. C. Aiello, J. Doyle, and S. C. Shapiro, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR '96)*, pages 529–541. San Francisco, CA, Morgan Kaufmann, 1996.

[BFN96] Peter Baumgartner, Ulrich Furbach, and Ilkka Niemelä. Hyper tableaux. In L.M. Pereira J.J. Alferes and E. Orlowska, editors, *Logics in Artificial Intelligence (JELIA '96)*, LNCS 1126, pages 1–17. Springer, 1996.

[BL93] Stefan Brass and Udo W. Lipeck. Bottom-up query evaluation with partially ordered defaults. In Stefano Ceri, Katsumi Tanaka, and Shalom Tsur, editors, *Deductive and Object-Oriented Databases, Third Int. Conf., (DOOD'93)*, number 760 in LNCS, pages 253–266, Berlin, 1993. Springer.

[DFN97] J. Dix, U. Furbach, and A. Nerode, editors. *Logic Programming and Nonmonotonic Reasoning*, LNAI 1265, Berlin, 1997. Springer.

[Dix95] Jürgen Dix. Semantics of Logic Programs: Their Intuitions and Formal Properties. An Overview. In Andre Fuhrmann and Hans Rott, editors, *Logic, Action and Information – Essays on Logic in Philosophy and Artificial Intelligence*, pages 241–327. DeGruyter, 1995.

[EG93] Thomas Eiter and Georg Gottlob. Complexity aspects of various semantics for disjunctive databases. In *Proc. of the Twelfth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'93)*, pages 158–167, 1993.

[EG96] Thomas Eiter and Georg Gottlob. Mächtigkeit von Logikprogrammierung über Datenbanken. *KI*, 3:32–39, 1996.

[EGM94] Thomas Eiter, Georg Gottlob, and Heikki Mannila. Adding disjunction to datalog. In *Proc. of the Thirteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'94)*, pages 267–278, 1994.

[ELM$^+$97] T. Eiter, N. Leone, C. Mateis, G. Pfeifer, and F. Scarcello. A Deductive System for Non-Monotonic Reasoning. In J. Dix, U. Furbach, and A. Nerode, editors, *Logic Programming and Non-Monotonic Reasoning, Proceedings of the Fourth International Conference*, LNAI 1265, pages 363–374, Berlin, June 1997. Springer.

[GL88] Michael Gelfond and Vladimir Lifschitz. The Stable Model Semantics for Logic Programming. In R. Kowalski and K. Bowen, editors, *5th Conference on Logic Programming*, pages 1070–1080. MIT Press, 1988.

[GL90] Michael Gelfond and Vladimir Lifschitz. Logic Program with Classical Negation. In David H.D. Warren and Peter Szeredi, editors, *Proceedings of the 7th Int. Conf. on Logic Programming*, pages 579–597. MIT, June 1990.

[GPP88] Michael Gelfond, Halina Przymusinska, and Teodor Przymusinski. The extended Closed World Assumption and its Relationship to parallel Circumscription. In *Proceedings 7th Symposion on Principles of Database Systems*, 1988.

[Lip79] W. Lipski, Jr. On semantic issues connected with incomplete information databases. *ACM Transactions on Database Systems*, 4:262-296, 1979.

[LMR92] Jorge Lobo, Jack Minker, and Arcot Rajasekar. *Foundations of Disjunctive Logic Programming.* MIT-Press, 1992.

[McC80] John McCarthy. Circumscription: A Form of Nonmonotonic Reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[Min82] Jack Minker. On indefinite databases and the closed world assumption. In *Proceedings of the 6th Conference on Automated Deduction, New York*, pages 292–308, Berlin, 1982. Springer.

[Min96] Jack Minker. Logic and databases: A 20 year retrospective. In Dino Pedreschi and Carlo Zaniolo, editors, *Proceedings of the International Workshop on Logic in Databases (LID)*, LNCS 1154, pages 3–58. Springer, Berlin, 1996.

[Nie96a] Ilkka Niemelä. Implementing circumscription using a tableau method. In W. Wahlster, editor, *Proceedings of the European Conference on Artificial Intelligence*, pages 80–84, Budapest, Hungary, August 1996. John Wiley.

[Nie96b] Ilkka Niemelä. A tableau calculus for minimal model reasoning. In P. Miglioli, U. Moscato, D. Mundici, and M. Ornaghi, editors, *Proceedings of the Fifth Workshop on Theorem Proving with Analytic Tableaux and Related Methods*, pages 278–294, Terrasini, Italy, May 1996. LNAI 1071, Springer-Verlag.

[Prz95] Teodor Przymusinski. Static Semantics For Normal and Disjunctive Logic Programs. *Annals of Mathematics and Artificial Intelligence*, 14:323–357, 1995.

[Prz98] Teodor C. Przymusinski. Autoepistemic logic of knowledge and beliefs. *Artificial Intelligence*, 95:115–154, 1998.

[Sei94] Dietmar Seipel. An efficient computation of the extended generalized closed world assumption by support-for-negation sets. In *Proc. Int. Conf. on Logic Programming and Automated Reasoning (LPAR'94)*, number 822 in LNAI, pages 245–259, Berlin, 1994. Springer.

[SS94] Chiaki Sakama and Hirohisa Seki. Partial Deduction of Disjunctive Logic Programs: A Declarative Approach. In *Logic Program Synthesis and Transformation – Meta Programming in Logic*, LNCS 883, pages 170–182, Berlin, 1994. Springer.

[YY93] Jia-Huai You and Li-Yan Yuan. Autoepistemic Circumscription and Logic Programming. *Journal of Automated Reasoning*, 10:143–160, 1993.

Preferred Answer Sets for Extended Logic Programs

Gerhard Brewka
Universität Leipzig
Institut für Informatik
Augustusplatz 9/10, D-04109 Leipzig
Germany
brewka@informatik.uni-leipzig.de

Thomas Eiter
Justus-Liebig-Universität Gießen
Institut für Informatik
Arndtstraße 2, D-35392 Gießen
Germany
eiter@informatik.uni-giessen.de

Abstract

In this paper, we extend Gelfond and Lifschitz's answer set semantics to prioritized extended logic programs. In such programs, an ordering on the program rules is used to express preferences. We show how this ordering can be used to define preferred answer sets and thus to increase the set of consequences of a program. We define a strong and a weak notion of preferred answer sets. The first takes preferences more seriously and satisfies reasonable principles for priority handling in rule based systems, while the second guarantees the existence of a preferred answer set for consistent programs. We show that strong preference on answer sets does not add on the complexity of the principal reasoning tasks, and weak preference leads only to a mild increase in complexity.

1 Introduction

Preferences among default rules play an important role in applications of nonmonotonic reasoning. One source of preferences that has been studied intensively is specificity [27, 35, 36]. In case of a conflict between defaults we tend to prefer the more specific one since this default provides more reliable information.

Specificity is an important source of preferences, but not the only one. In the legal domain it may, for instance, be the case that a more general rule is preferred since it represents federal law as opposed to state law [29]. In these cases preferences may be based on some basic principles regulating how conflicts among legal rules are to be resolved. Also in other application domains, like model based diagnosis, configuration or decision making, preferences play a fundamental role and their relevance is well-recognized.

Prioritized versions for most of the existing nonmonotonic formalisms have been proposed, e.g., prioritized circumscription [16], hierarchic autoepistemic logic [19], prioritized default logic [24, 5, 1], prioritized theory revision [3, 25], or prioritized abduction [11]. Somewhat surprisingly, preferences have received less attention in logic programming. This may be explained by the fact that for a long period, logic programming was mainly conceived as a logical paradigm for declarative programming. However, in the recent past, it has become evident that logic programming can also serve as a powerful framework for knowledge representation, cf. [13, 2]. If logic programming wants to successfully stand this challenge, it must provide features like preference handling which have been recognized as indispensable in the context of knowledge representation.

In this paper we consider extended logic programs under answer set semantics supplied with priority information, which is given by a supplementary strict partial ordering of the rules. This additional information is used to solve potential conflicts; that is, we want to conclude more than in standard answer set semantics. On the other hand, unless the program with the given preference information is unsatisfiable (in a sense to be made precise) we want to conclude only literals that are contained in at least one answer set. The best way to achieve these goals is to use the preferences on rules for selecting a subset of the answer sets, which we call the *preferred answer sets*.

Our approach is based on the following two main ideas. The first is that the application of a rule with nonmonotonic assumptions means to jump to a conclusion, and this conclusion is yet another assumption which has to be used globally in the program for the issue of deciding whether a rule is applicable or not. The second is that the rules must be applied in an order compatible with the priority information. We take this to mean that a rule is applied *unless it is defeated via its assumptions by rules of higher priorities*. This view is new and avoids the unpleasant behavior which is present with other approaches.

The paper is organized as follows. The next section recalls the definitions of extended logic programs and introduces basic notations. Section 3 introduces two basic principles for preference handling in rule based nonmonotonic systems, reviews some approaches to prioritized default logic and demonstrates that they fail to satisfy the principles. In Section 4, we then present our approach, by introducing the concept of preferred answer sets. We demonstrate the approach on a number of examples, and investigate in Section 5 its properties. The existence of preferred answer sets for consistent programs is not guaranteed. We define in Section 6 also a weak notion of preferred answer sets. Intuitively, the strong notion of answer sets takes preferences more seriously, which may lead to a situation in which the priorities are incompatible with the answer set condition; this is avoided in the weaker notion. In Section 7, we address complexity issues. Section 8 discusses further related work and concludes.

Due to space limitations all proofs are left out. They are contained in the full version of the paper which is available from the authors.

2 Preliminaries and Notation

We assume that the reader is familiar with the basic concepts of logic programming; see [21] for background. In the present paper, we focus on extended logic programs, which have two kinds of negation, as described in [15].

As usual, let $\mathcal{L}$ be an underlying countable first-order language. Unless stated otherwise, $\mathcal{L}$ is the language generated by the program or rule base under consideration.

A *rule* r is a formula

$$c \leftarrow a_1, \ldots, a_n, \text{not } b_1, \ldots, \text{not } b_m \qquad (1)$$

where the a_i, b_j and c are classical literals, i.e., either positive atoms or atoms preceded by the classical negation sign $\neg$. We denote by *head*(r) the head of rule r. The symbol not denotes negation by failure (*weak negation*), while $\neg$ denotes *strong negation* (sometimes called explicit negation). We will call $a_1, \ldots, a_n$ the *prerequisites* of the rule and use *pre*(r) to denote the set of prerequisites of r. A rule is called *prerequisite-free*, if $n = 0$.

A *rule base* R is a (possibly infinite) collection of rules; an *extended logic program* (*logic program* or *program*, for short) P is a finite rule base.

As usual, a rule (resp. rule base, program) is *ground*, if no variable occurs in it; a rule base (resp., program) is prerequisite-free, if all rules in it are prerequisite-free.

For a rule base R, we denote by R^* the ground instantiation of R over the Herbrand universe of the language $\mathcal{L}$. Moreover, we denote by *Lits* the set of all classical ground literals of $\mathcal{L}$.

We say a rule r of the form (1) is *defeated by a literal* ℓ, if $\ell = b_i$ for some $i \in \{1, \ldots, m\}$, and we say it is *defeated by a set of literals* X, if X contains a literal that defeats r.

Let us recall the definition of the answer set semantics for extended logic programs [15]. Answer sets are defined in analogy to stable models [14], but taking into account that atoms may be preceded by classical negation.

Definition 2.1 *Let R be a collection of ground rules, and let $X \subseteq Lits$ be a set of ground literals. The reduct of R with respect to X (for short, X-reduct of R), denoted R^X, is the collection of rules resulting from R by*

- *deleting each rule which is defeated by X, and*
- *deleting all weakly negated literals from the remaining rules.*

This reduction of R is often called *Gelfond-Lifschitz reduction*, after its inventors [14].

Definition 2.2 *Let R be a collection of ground rules without weak negation. Then, $Cn(R)$ denotes the smallest set $S \subseteq Lits$ of ground literals such that*

1. *S is closed under R, i.e., for any rule $a \leftarrow a_1, \ldots, a_n$ in R, if $a_1, \ldots, a_n \in S$, then $a \in S$; and*
2. *S is logically closed, i.e., either S is consistent or $S = Lits$.*

Definition 2.3 *Let R be a collection of ground rules. Define an operator $\gamma_R(X)$ on the sets X of ground literals as follows:*

$$\gamma_R(X) = Cn(R^X).$$

Then, a set X of ground literals is an answer set of R iff $X = \gamma_R(X)$.

The collection of answer sets of R is denoted by $\mathcal{AS}(R)$. For an arbitrary logic program P, the collection of answers sets $\mathcal{AS}(P)$ is given by $\mathcal{AS}(P) = \mathcal{AS}(P^*)$.

A ground literal L is a consequence of a program P under answer set semantics, denoted $P \models L$, iff L is contained in all answer sets of P. For more on answer sets, consult [15].

3 Why Existing DL-Approaches Do Not Work

Different prioritized versions of Reiter's default logic [31] have been proposed in the literature, e.g. [24, 5, 1, 32, 9]. We will show that all of them suffer from weaknesses and thus cannot serve –via the standard translation from extended logic programs to default theories– as a satisfactory specification of preferred answer sets.

3.1 Principles for priorities

Before discussing these approaches, we want to formulate two principles which as we believe should be satisfied by any system which is based on prioritized defeasible rules. Since we want our principles to cover different approaches like default logic, prioritized logic programs, etc. we use in this section the generic terms *belief set* (for extension, answer set, ...) and *prioritized theory* (for prioritized default theory, prioritized logic program,...) in our formulations.

The first principle can be viewed as a meaning postulate for the term "preference" and states a minimal requirement for preference handling in rule based systems:

Principle I.
Let B_1 and B_2 be two belief sets of a prioritized theory $(T, <)$ generated by the (ground) rules $R \cup \{d_1\}$ and $R \cup \{d_2\}$, where $d_1, d_2 \notin R$, respectively. If d_1 is preferred over d_2, then B_2 is not a (maximally) preferred belief set of T.

In this context, a rule r is said to *generate* a belief set B, if the prerequisites of r are in B and B does not defeat r. It is hard to see how the use of the term "preference among rules" could be justified in cases where Principle I is violated.

The second principle is related to relevance. It tries to capture the idea that the decision whether to believe a formula p or not should depend on the priorities of rules contributing to the derivation of p only, not on the priorities of rules which become applicable when p is believed:

Principle II.
Let B be a preferred belief set of a prioritized theory $(T, <)$, r a (ground) rule such that at least one prerequisite of r is not in B. Then B is a preferred belief set of $(T \cup \{r\}, <')$ whenever $<'$ agrees with $<$ on priorities among rules in T.

Thus, adding a rule which is not applicable in a preferred belief set can never render this belief set non-preferred unless new preference information changes preferences among some of the old rules (e.g. via transitivity). In other words, a belief set is not blamed for not applying rules which are not applicable.

We will see that each of the existing treatments of preferences for default logic, described in [24, 5, 1, 32, 9], violates one of these principles.

3.2 Control of Reiter's quasi-inductive definition

The first group of proposals [24, 5, 1] uses preferences to control the quasi-inductive definition of extensions [31]: in each step of the generation of extensions the defaults with highest priority whose prerequisites have already been derived are applied. This leads to situations where the preferred extensions do not take seriously what they believe. It may be the case that a less preferred default is applied although the prerequisite of a conflicting, more preferred default is believed in a preferred extension. As we will see, this leads to the violation of Principle I.

The mentioned approaches differ in technical detail. We do not want to present the exact definitions here. Instead, we will illustrate the difficulties using an example for which all three approaches obtain the same result.

Example 3.1 Assume we are given the following default theory:[1]

(1) $a : b/b$
(2) $true : \neg b/\neg b$
(3) $true : a/a$

Assume further that (1) is preferred over (2) and (2) over (3). This default theory has two Reiter extensions, namely $E_1 = Th(\{a, b\})$, which is generated by rules (1) and (3), and $E_2 = Th(\{a, \neg b\})$, which is generated by rules (2) and (3). The single preferred extension in the approaches mentioned above is E_2. The reason is that the prerequisite of (2) is derived before the prerequisite of (1) in the construction of the extension. The approaches thus violate Principle I. ▮

3.3 Rintanen's approach

Rintanen [32] uses a total order on (normal) defaults to induce a lexicographic order on extensions.

Call a default rule $r = a{:}b/b$ *applied* in a set of formulas E (denoted $r \in appl(E)$), if a and b are in E. An extension E is then preferred over extension E', if and only if there is a default $r \in appl(E) \setminus appl(E')$ satisfying the following condition: if r' is preferred over r and $r' \in appl(E')$, then $r' \in appl(E)$.

Unfortunately, also this approach leads to counterintuitive results and to a violation of our principles.

Example 3.2 Consider the following default theory, which is similar to the one in Example 3.1:

(1) $a : b/b$
(2) $true : \neg a/\neg a$
(3) $true : a/a$

[1]We assume that the reader knows the basic concepts of standard default logic [31]; informally, a default rule $a{:}b/c$ corresponds to the clause $c \leftarrow$ not$\neg b, a$.

Again (1) is preferred over (2), and (2) over (3). The default theory has two Reiter extensions, namely $E_1 = Th(\{\neg a\})$ and $E_2 = Th(\{a, b\})$. Intuitively, since the decision whether to believe a or not depends on (2) and (3) only, and since (2) is preferred over (3), we would expect to conclude $\neg a$, in other words, to prefer E_1.

Rintanen, however, prefers E_2. The reason is that in E_2 default (1) is applied. Belief in a is thus accepted on the grounds that this allows us to apply a default of high priority. This is far from being plausible and amounts to wishful thinking. It is also easy to see that Principle II is violated: E_1 clearly is the single preferred extension of rules (2) and (3) in Rintanen's approach. Adding rule (1) which is not applicable in E_1 makes E_1 a non-preferred extension. ▮

4 Prioritized Programs and Preferred Answer Sets

In this section, we present our approach for incorporating priorities into extended logic programs. In this approach, priorities are specified like in other approaches by an ordering of the rules. This ordering will be used to test if an answer set is constructed by applying the rules in a proper order. However, and this is the salient point of our approach, the proper order of rule application will not be enforced during a quasi-inductive construction of an answer set, but will rather be ensured in a separate additional condition which requires a dual reconstruction of the answer set. Our approach derives from the following two underlying ideas:

(1) Applying rules with default negation means to jump to conclusions, and any such a conclusion has to be used globally throughout the program when the applicability of a rule is tested.

(2) Rules must be applied in an order compatible with the priorities. It appears that this is intuitively the case if each rule whose prerequisites are true and whose assumptions are not defeated by the rules with higher priority, is applied.

4.1 Prioritized extended logic programs

We start with the syntactical part of our framework.

Definition 4.1 *A prioritized rule base $\mathcal{R} = (R, <)$ is a pair where P is a rule base and $<$ a strict partial order on R. In particular, $\mathcal{R}$ is a prioritized (extended) logic program, if R is an (extended) logic program.*

Recall that a strict partial order is an irreflexive and transitive relation. The order $<$ is used to express preference information: $r_1 < r_2$ stands for "r_1 has higher priority than r_2".

We use here a partial order rather than a total order on the rules, which is appropriate for different reasons. The first reason is that in some scenarios it may be unwanted or even unnatural to specify an order between rules. A second reason is that, as rules represent their ground instances, we can pass easily from a program P with variables to its ground instantiation P^* while the intuitive meaning of priorities on rules is preserved. E.g., if we have rules such that $r_1 < r_2$, then by passing to the ground instances, the naturally induced partial order says that every instance of r_1 has higher priority than any instance of r_2, while no priorities are given between the instances of r_1 and of r_2, respectively.

If we adopt the view that rules should be applied one at a time, then a partial ordering $<$ on the rules is a representative of all possible refinements of $<$ to total orderings. However, not all total orderings are intuitively acceptable. It is natural to assume that in any totally ordered rule base, some rule has highest priority. This reflects the view of proceeding from most important to less important matters, and means that infinitely decreasing chains $r_1 > r_2 > r_3 > \ldots$ of rules are excluded. Formally, this amounts to the condition that the ordering $<$ on R is well-founded. Any such ordering is called a *well-ordering*.

Each total ordering of a finite set is trivially a well-ordering, while this is not true for infinite sets. It is well-known that each well-ordering $<$ of a set M corresponds by its order type to a unique ordinal number $ord(<)$, and thus to a (possibly transfinite) enumeration $m_0, m_1, \ldots, m_\alpha, \ldots$ of the elements in M, where α is an ordinal number such that $0 \leq \alpha < ord(<)$; we use the notation $(M, <) = \{m_\alpha\}_<$ for this. Moreover, every well-ordering $<$ induces on every nonempty subset $M' \subseteq M$ a unique well-ordering $<'$. For a background on well-orderings, see e.g. [34].

Definition 4.2 *Let $\mathcal{P} = (P, <)$ be a prioritized logic program. Then, the ground instantiation of $\mathcal{P}$ is the prioritized rule base $\mathcal{P}^* = (P^*, <^*)$ where $<^*$ is the relation on P^* satisfying $r_1^* <^* r_2^*$ iff r_1^* and r_2^* are instances of rules r_1 and r_2 in P, respectively, such that $r_1 < r_2$.*

Note that in general, the relation $<^*$ in $(P^*, <^*)$ is not a partial ordering; this happens if the priorities $<$ are not consistent, in the sense that ground instances of rules may lead to a priority conflict. E.g., if we have rules $r_1 : p(x) \leftarrow q(x, a)$ and $r_2 : p(b) \leftarrow q(y, x)$ where $r_1 < r_2$, then the rule $r_3 : p(b) \leftarrow q(b, a)$ is a common instance of r_1 and r_2, which raises the contradicting priority $r_3 <^* r_3$. Such contradictions can be effectively recognized from $(P, <)$, however, and we thus may assume that $<^*$ is in fact a partial order. Moreover, it is easy to avoid such conflicts, by tagging rules with dummy literals such that common rule instances are not possible.

Definition 4.3 *A full prioritization of any ordered ground rule base $\mathcal{R} = (R, <)$ is any pair $\mathcal{R}' = (R, <')$ where $<'$ is a well-ordering on R compatible with $<$, i.e., $r_1 < r_2$ implies $r_1 <' r_2$, for all $r_1, r_2 \in R$. By $\mathcal{FP}(\mathcal{R})$ we denote the collection of all full prioritizations of $\mathcal{R}$. We say that $\mathcal{R}$ is fully prioritized, if $\mathcal{FP}(\mathcal{R}) = \{\mathcal{R}\}$, i.e., $\mathcal{R}$ coincides with its unique full prioritization.*

Every well-ordering $<'$ which refines $<$ is compatible with the priority specification, and if some answer set is acceptable under $<'$, it should be acceptable under $<$ as well; we thus will define the concept of preferred answer set under this credulous view of priorities.

4.2 Preferred answer sets

We will first formulate the criterion which an answer set must satisfy in order to be preferred, given a fully prioritized rule base $\mathcal{R} = (R, <)$.

Recall that in the definition of answer sets, the case of rules with weak negation is reduced to a particularly easy special case, namely to rules without weak negation. Similarly, there is a special case where it is particularly easy to check for an answer set A whether a full prioritization $(R, <)$ was taken into account adequately: ground rule bases R without prerequisites, i.e. collections where the bodies of all rules contain only weakly negated literals. In this case, we simply have to check whether each rule in R whose head is *not* in A is *defeated* by the consequences of applied rules of higher priority.

To model this, we associate with each fully prioritized rule base $\mathcal{R}$ of prerequisite-free ground rules an operator $C_{\mathcal{R}} : 2^{Lits} \to 2^{Lits}$. An answer set A satisfies the priorities, in case it is a fixpoint of $C_{\mathcal{R}}$. In the construction of $C_{\mathcal{R}}(A)$, rules are applied in the order of their priorities. In each step, the head of the current rule r is added to the collected literals, if (1) r is not defeated by literals collected so far, and (2) it is not the case that $head(r) \in A$ yet r is defeated in A. Note that condition (2) is necessary to avoid situations where a rule defeated in A nevertheless increases the priority of a different derivation of its head through another rule. The formal definition is as follows.

Definition 4.4 *Let $\mathcal{R} = (R, <)$ be a fully prioritized rule base of prerequisite-free ground rules, S a set of literals, and let $\mathcal{R} = \{r_\alpha\}$. We define the sequence S_α, $0 \le \alpha < ord(<)$, of sets $S_\alpha \subseteq Lits$ as follows:*

$S_\alpha =$

$\bigcup_{\beta<\alpha} S_\beta$ *if* r_α *is defeated by* $\bigcup_{\beta<\alpha} S_\beta$ *or* $c \in S$ *and* r_α *is defeated by* S,

$\bigcup_{\beta<\alpha} S_\beta \cup \{c\}$ *otherwise,*

where $c = head(r_\alpha)$. The set $C_{\mathcal{R}}(S)$ is the smallest set of ground literals such that

(i) $\bigcup_{\alpha<ord(<)} S_\alpha \subseteq C_{\mathcal{R}}(S)$, *and*

(ii) $C_{\mathcal{R}}(S)$ *is logically closed.*

Note that for $\alpha = 0$, $\bigcup_{\beta<\alpha} S_\beta = \emptyset$ holds. Moreover, for each successor ordinal $\alpha + 1$, the definition of $S_{\alpha+1}$ can be simplified by replacing $\bigcup_{\beta<\alpha+1} S_\beta$ with S_α; the above definition is uniform and more succinct, however. The sequence S_α monotonically increases and converges to $\bigcup_{\alpha<ord(<)} S_\alpha$.

$C_{\mathcal{R}}$ is not meant to return consequences of R. It may well be the case that a rule r_α is applied in the production of $C_{\mathcal{R}}(S)$ although the rule is later defeated by some less preferred rule r_β, i.e., $\alpha < \beta$. However, if some answer set A of R is a fixpoint of $C_{\mathcal{R}}$, then we can be sure that all preferences in $<$ were taken into account adequately, that is, a rule whose head is not in A is defeated by a more preferred rule applied in A. We therefore define preferred answer sets as follows:

Definition 4.5 *Let $\mathcal{R} = (R, <)$ be a fully prioritized rule base of prerequisite-free ground rules, and let A be an answer set of R. Then A is the preferred answer set of $\mathcal{R}$, if and only if $C_{\mathcal{R}}(A) = A$.*

Let us check that the use of the term "*the*" preferred answer set is justified in Definition 4.5.

Lemma 4.1 *Let $\mathcal{R} = (R, <)$ be a fully prioritized rule base of prerequisite-free ground rules, $\mathcal{R}$ possesses at most one preferred answer set.*

In the case of a rule base R of arbitrary ground clauses, we perform a reduction which can be viewed as dual to the Gelfond-Lifschitz reduction: given a set of ground literals A, we eliminate rules whose *prerequisites* are not in A and eliminate all prerequisites from the remaining rules. This yields a rule base of prerequisite-free ground rules $\mathcal{R}' = (R', <')$, together with an ordering $<'$ that is inherited from $\mathcal{R}$. We then can check whether a given answer set is a fixpoint of $C'_{\mathcal{R}}$.

Intuitively, our construction amounts to guessing provable prerequisites and checking whether the assumption that exactly these prerequisites hold is possible under the prioritized interpretation of rules.

Definition 4.6 *Let $(R, <) = \{r_\alpha\}_<$ be a fully prioritized set of ground rules, and let X be a set of ground literals. Let ${}^X\mathcal{R} = ({}^XR, {}^X<)$ be the fully prioritized set of ground rules such that ${}^XR$ is the set of rules obtained from R by*

1. *deleting every rule having a prerequisite ℓ such that $\ell \notin X$, and*

2. *removing from each remaining rule all prerequisites,*

and ${}^X\!<$ is inherited from $<$ by the map $f : {}^XR \longrightarrow R$, i.e., $r'_1 \;{}^X\!< r'_2$ iff $f(r'_1) < f(r'_2)$, where $f(r')$ is the $<$-least rule r_α in R such that r' results from r by step 2.

It is easily seen that ${}^X\!<$ is indeed a well-ordering on ${}^XR$. The definition of ${}^X\!<$ may look somewhat involved, but we have to respect a possible clash of rule priorities due to Step 2 of the reduction.

With these definitions, we are prepared for generalizing the notion of preferred answer sets to arbitrary programs:

Definition 4.7 *Let A be a set of ground literals. A is a preferred answer set of a fully prioritized set of ground rules $\mathcal{R} = (R, <)$, if A is a preferred answer set of ${}^A\mathcal{R}$.*

A set of ground literals A is a preferred answer set of a prioritized logic program $\mathcal{P}$, if A is a preferred answer set for some $\mathcal{R} \in \mathcal{FP}(\mathcal{P}^*)$.

The collection of all preferred answer sets of $\mathcal{R}$ (resp. $\mathcal{P}$) is denoted by $\mathcal{PAS}(\mathcal{R})$ (resp. $\mathcal{PAS}(\mathcal{P})$).

Notation. For convenience, we introduce an operator $\lambda_\mathcal{R}$ as follows: $\lambda_\mathcal{R}(X) = C_{{}^X\mathcal{R}}(X)$. An answer set is then obviously preferred if and only if it is a fixpoint of $\lambda_\mathcal{R}$. In other words, A is a preferred answer set iff it is a fixpoint of Gelfond and Lifschitz's operator γ_R (see Definition 2.3) and a fixpoint of $\lambda_\mathcal{R}$.

Here is a first simple example. More examples will be discussed in the following sections. For simplicity, unless specified otherwise we will assume that the rules in each example are ordered by the numbering of the rules, i.e., rules with lower numbers are preferred over those with higher numbers.

Example 4.1 Let us consider the classical Tweety example, but impose some preferences on the rules as in Section 1:

$$\begin{array}{lr}
(1) & peng(tweety) \leftarrow \\
(2) & bird(tweety) \leftarrow \\
(3) & \neg flies(x) \leftarrow \text{not } flies(x), peng(x) \\
(4) & flies(x) \leftarrow \text{not } \neg flies(x), bird(x)
\end{array}$$

As mentioned above, P has two answer sets: $A_1 = \{peng(tweety), bird(tweety), \neg flies(tweety)\}$ and $A_2 = \{peng(tweety), bird(tweety), flies(tweety)\}$.

Notice that P^* contains besides (1) and (2) the rules (3) and (4) instantiated with $x = tweety$, and the inherited ordering $<^*$ on P^* is already a well-ordering. Thus, $\mathcal{P}^* = (P^*, <^*)$ is fully prioritized.

Let us first check whether A_1 is a preferred answer set. First, we have to determine the dual reduct ${}^{A_1}\mathcal{P}^*$. It contains the reduct of all four rules. Now, let us determine $\lambda_{\mathcal{P}^*}(A_1)$. It is not difficult to see that A_1 is indeed reconstructed in this process; consequently, the answer set A_1 is preferred.

On the other hand, let us compute $\lambda_{\mathcal{P}^*}(A_2)$. The dual reducts ${}^{A_2}\mathcal{P}^*$ and ${}^{A_1}\mathcal{P}^*$ coincide, and $\lambda_{\mathcal{P}^*}(A_2) = A_1$. Hence, A_2 is not preferred. ▮

5 Properties of Preferred Answer Sets

In this section, we show that the preferred answer set approach has several appealing properties, which make it particularly attractive.

First of all, we show that the problems with existing DL-approaches discussed in Section 3 are resolved in our approach.

Example 5.1 Consider the logic programming version of the example we used in Section 3.2 to demonstrate the violation of Principle I in the first group of DL-proposals:

$$\begin{array}{lr}
(1) & b \leftarrow \text{not} \neg b, a \\
(2) & \neg b \leftarrow \text{not } b \\
(3) & a \leftarrow \text{not} \neg a
\end{array}$$

The program has two answer sets: $A_1 = \{a, b\}$ and $A_2 = \{a, \neg b\}$. As intended, A_2 is not preferred since $\lambda_\mathcal{P}(A_2) = A_1$. The single preferred answer set is A_1. ▮

Example 5.2 Let us turn to the problematic example for Rintanen's approach. Also this case is handled adequately:

$$\begin{array}{lr}
(1) & b \leftarrow \text{not} \neg b, a \\
(2) & \neg a \leftarrow \text{not } a \\
(3) & a \leftarrow \text{not} \neg a
\end{array}$$

We obtain the two answer sets $A_1 = \{\neg a\}$ and $A_2 = \{a, b\}$. The answer set A_1 is preferred: ${}^{A_1}\mathcal{P}$ contains merely rules (2) and (3), and A_1 is fixpoint of $\lambda_\mathcal{P}$. On the other hand, A_2 is not preferred since $\lambda_\mathcal{P}(A_2) = \{b, \neg a\}$. This is exactly what we expect. ▮

The desired behavior of our approach on the problematic examples for the other approaches is not incidental. In fact, it satisfies the Principles I and II which we have introduced above, as we demonstrate next.

We start by showing that our approach captures a natural and intuitive idea about preferences. Recall that a rule r is applied in A exactly if $pre(r) \subseteq A$ and r is not defeated in A.

Proposition 5.1 *Let $\mathcal{R} = (R, <)$ be a fully prioritized rule base of ground rules, and let $A \in \mathcal{AS}(R)$. Then, A is a preferred answer set of $\mathcal{R}$, if and only if for each rule $r \in R$ such that $pre(r) \subseteq A$ and $head(r) \notin A$, there is a rule $r' \in R$ generating A such that $r' < r$ and $head(r')$ defeats r.*

We can also show that the behavior observed in Examples 5.1 and 5.2 is not incidental: our approach actually satisfies the desired principles introduced in Section 3.1.

Proposition 5.2 *The preferred answer set approach satisfies Principles I and II as described in Section 3.1.*

The following property, which is easily verified, confirms that our reduction of partial orders to full prioritizations works as intended.

Proposition 5.3 *Let $<_1$ and $<_2$ be partial orders on a program P such that $<_2$ refines $<_1$, i.e., $<_1 \subseteq <_2$. Then, $\mathcal{PAS}(P, <_2) \subseteq \mathcal{PAS}(P, <_1)$.*

Thus, as desired, removing priorities between rules does not affect a preferred answer set of a program and adding additional preference information can never lead to the loss of a conclusion.

Proposition 5.4 *Let R be a collection of ground rules, and let $A \in \mathcal{AS}(R)$ be an answer set. Then, there exists some well-ordering $<$ of R such that $A \in \mathcal{PAS}(\mathcal{R})$ where $\mathcal{R} = (R, <)$.*

An immediate consequence of this property is that prioritized logic programs are a conservative generalization of extended logic programs.

Corollary 5.5 *Let $\mathcal{P} = (P, <)$ without priorities, i.e., $<$ is empty. Then, $\mathcal{PAS}(\mathcal{P}) = \mathcal{AS}(\mathcal{P})$.*

Another property is that an inconsistent answer set is insensitive to any rule ordering.

Proposition 5.6 *Let $\mathcal{P} = (P, <)$ be a prioritized program such that $\mathcal{FP}(\mathcal{P}) \neq \emptyset$. If $A = Lits$ is an answer set of $\mathcal{P}$, then $Lits$ is the unique preferred answer set of $\mathcal{P}$.*

Preferred answer sets are not always unique, even for totally ordered programs.

Example 5.3 Consider the following program:

(1) $b \leftarrow \text{not}\,\neg b, a$
(2) $c \leftarrow \text{not}\ b$
(3) $a \leftarrow \text{not}\ c$

This program has two answer sets: $A_1 = \{a, b\}$ and $A_2 = \{c\}$. Both are preferred. ▮

On the other hand, it is not difficult to see that there are also programs that have answer sets but for which no preferred answer set exists.

Example 5.4 Consider the following program:

(1) $c \leftarrow \text{not}\ b$
(2) $b \leftarrow \text{not}\ a$

The single answer set of this program is $A = \{b\}$. However, $C_{\mathcal{A}\mathcal{P}}(A) = \{c, b\}$ and thus A is not preferred. ▮

Two views seem possible here: one might say that this is exactly what is expected in a situation like this; the priorities cannot be satisfied and thus the prioritized program should be considered inconsistent. This corresponds to a rather strict interpretation of the meaning of priorities.

However, one might also argue that the preference information should help us in selecting an answer set from all existing answer sets, but never render an otherwise consistent program inconsistent. In the next section, we show how this view of relaxing priority information can be implemented in our approach.

6 Weakly Preferred Answer Sets

We now propose a relaxation of our approach that gives us exactly the preferred answer sets whenever they exist and some approximation, called *weakly preferred answer sets*, in the other cases. Of course, we want weakly preferred answer sets to be answer sets. Note that these requirements are incompatible with our two principles as simple examples demonstrate (to see that Principle II is incompatible add the rule $a \leftarrow c$ with highest priority to the program of Example 5.4; an example demonstrating violation of Principle I is contained in the full paper). We trade in this section satisfaction of our principles for consistency of the semantics.

The basic idea is to consider in the comparison of answer sets the degree to which preferences are violated. Intuitively, the degree depends on how much of the current priority information has to be changed in order to obtain a preferred answer set. To measure this appropriately, we need a suitable notion of distance between two well-orderings of the same set.

With an eye on our application, we may consider the distance from a well-ordering $<_1$ of a set M to another well-ordering $<_2$ of M as the number of pairs m, m' such that $m <_1 m'$ but $m' <_2 m$, i.e., the number of violations (or *inversions*) of the ordering $<_2$ in $<_1$. For example, consider $M = \{a, b, c, d\}$ and the two orderings $c <_1 a <_1 d <_1 b$ and $a <_2 b <_2 c <_2 d$; then the pairs (c, a), (c, b), and (d, b) are the inversions of $<_2$ in $<_1$. What we want to do is basically count these inversions. This leads us to the following definition.

Definition 6.1 *Let $<_1$ and $<_2$ be well-orderings of the set M with corresponding enumerations $(M, <_1) = \{r_\alpha\}_{<_1}$ and $(M, <_2) = \{s_\beta\}_{<_2}$, respectively. For $\beta < ord(<_2)$ let $V_\beta(<_1, <_2)$ denote the well-ordering induced by $<_1$ on the set*

$$\{r_\alpha \mid r_\alpha <_1 s_\beta, s_\beta <_2 r_\alpha\} \subseteq M.$$

Then, the distance from $<_1$ to $<_2$, denoted $d(<_1, <_2)$, is defined as the ordinal

$$d(<_1, <_2) = \sum_{\beta < ord(<_2)} ord(V_\beta(<_1, <_2,)).$$

The definition of $d(<_1, <_2)$ involves ordinal arithmetic, which may be less familiar to the reader; see e.g. [34] for a detailed treatment. Informally, addition $\alpha + \beta$ of ordinals amounts to appending an ordering of type β to an ordering of type α. Note that ordinal arithmetic generalizes familiar integer arithmetic to the infinite, and has there different properties. E.g., $1 + \omega = \omega$, which is different from $\omega + 1$; hence, addition of ordinals is not commutative in general.

The value of $d(<_1, <_2)$ is indeed well-defined, as every element of the infinite sum is an ordinal and the sequence is well-founded; for such sums, the result is an ordinal [34]. Intuitively, $ord(V_\beta(<_1, <_2))$ is the number of violations of the priority of the element s_β which is at position β in $<_2$, and $d(<_1, <_2)$ counts in a way all these violations. Counting them is easy if the underlying set M is finite, while it is not in the infinite case. The problem is that in the latter case, the order in which a collection of ordinals is summed matters in general; the result may even be undefined. Our selection in the definition of $d(<_1, <_2)$ is driven by our application, and sums the violations in order of decreasing position with respect to the violated ordering $<_2$.

Clearly, $d(<, <) = 0$ holds for any well-ordering $<$, while symmetry does not hold in general. If we wish, we can make the distance symmetric by choosing the smaller of $d(<_1, <_2)$ and $d(<_2, <_1)$. Or, we cast $d(<, <')$ to its associated cardinal; this is sufficient to make the distance symmetric for our application. In particular, for finite distance values, symmetry is automatically given.

Proposition 6.1 *Let $<_1, <_2$ be well-orderings of the same set M such that $d(<_1, <_2) < \omega$ is finite. Then, d is symmetric on $(<_1, <_2)$, i.e., $d(<_2, <_1) = d(<_1, <_2)$.*

Corollary 6.2 *On the full prioritizations of a ground rule base, the distance measure*

$$d'(<_1, <_2) = card(d(<_1, <_2)),$$

where $card(\alpha)$ is the cardinal associated with the ordinal α, is symmetric.

Note that using d' is particularly useful, if we want to avoid a fine-grained distinction of priority violation and we are only interested whether a finite or infinite number of priority violations is present; $d'(<_1, <_2)$ is either finite or $\aleph_0$, the smallest uncountable cardinal.

It can be checked that d verifies all axioms of a standard metric in the finite case, i.e., $d(<, <) = 0$, $d(<_1, <_2) = d(<_2, <_1)$, and $d(<_1, <_3) \leq d(<_1, <_2) + d(<_2, <_3)$.

With the above notion of distance from one well-ordering to another, we can weaken the definition of preferred answer sets as follows.

Definition 6.2 *Let $\mathcal{R} = (R, <)$ be a partially ordered ground rule base, and let $A \in \mathcal{AS}(R)$. The preference violation degree of A in $\mathcal{R}$, denoted $pvd_{\mathcal{R}}(A)$, is the minimum distance possible from any full prioritization of $\mathcal{R}$ to any fully prioritized rule base $\mathcal{R}' = (R, <')$ such that A is a preferred answer set of $\mathcal{R}'$.*

Moreover, let $pvd(\mathcal{R}) = \min_{A \in \mathcal{AS}(R)} pvd_{\mathcal{R}}(A)$ be the preference violation degree of $\mathcal{R}$.

For any answer set A of a prioritized program $\mathcal{P} = (P, <)$, the preference violation degree of A in $\mathcal{P}$, denoted $pvd_{\mathcal{P}}(A)$, is defined as $pvd_{\mathcal{P}}(A) = pvd_{\mathcal{P}^*}(A)$, and the preference violation degree of $\mathcal{P}$, denoted $pvd(\mathcal{P})$, is $pvd(\mathcal{P}) = pvd(\mathcal{P}^*)$.

Note that $pvd_{\mathcal{R}}(A)$ is indeed well-defined. This is a consequence of Proposition 5.4, which says that for every answer set A there exists a total ordering of the rules such that A is preferred.

Based on the definition of preference violation degree, we formalize the concept of weakly preferred answer sets as follows:

Definition 6.3 *Let $\mathcal{P} = (P, <)$ be a prioritized logic program. Then, an answer set A of P is a weakly preferred answer set of $\mathcal{P}$ if and only if $pvd_{\mathcal{P}}(A) = pvd(\mathcal{P})$. By $\mathcal{WAS}(\mathcal{P})$ we denote the collection of all weakly preferred answer sets of $\mathcal{P}$.*

The following results are easily verified.

Proposition 6.3 *Let $\mathcal{P} = (P, <)$ be a prioritized logic program. If $\mathcal{PAS}(\mathcal{P}) \neq \emptyset$, then $\mathcal{PAS}(\mathcal{P}) = \mathcal{WAS}(\mathcal{P})$.*

Preferred answer sets are just those with preference violation degree zero.

Proposition 6.4 *Let $\mathcal{P} = (P, <)$ be a prioritized logic program. If $\mathcal{AS}(\mathcal{P}) \neq \emptyset$, then $\mathcal{WAS}(\mathcal{P}) \neq \emptyset$.*

In particular, whenever P has a unique answer set, this set will be weakly preferred.

7 Computational Complexity

In this section, we address the issue of computing preferred and weakly preferred answer sets, and analyze the complexity of the main computational tasks in this context.

We assume that the reader is familiar with the basic concepts of complexity theory; [18, 26] are good sources. For a background on complexity results in logic programming, refer to [33, 10, 8]. In our analysis, we focus on the case of finite propositional (ground) prioritized programs.

Due to space limitations we can only list the major results without proofs and thorough discussion. These can be found in the full paper.

Of course, since extended logic programs without priorities are a special case of prioritized programs, all lower complexity bounds for answer sets of extended logic programs carry over to preferred answer sets. In particular, by the complexity results on the stable models of a general propositional logic program P [22, 23, 4] and the obvious correspondence between the preferred answer sets of the prioritized program $\mathcal{P} = (P, <)$ where $<$ is empty (cf. Corollary 5.5) and the stable models of P, we obtain that deciding whether a prioritized program has a preferred answer set is NP-hard, and, moreover, that inference of a literal from all preferred answer sets of a prioritized program is a co-NP-hard problem.

Natural questions are now whether preferred answer sets are harder than answer sets in general and, moreover, how the complexity is affected if we have programs with a particular priority ordering. We have proven the following theorem:

Theorem 7.1 *Let $\mathcal{P} = (P, <)$ be a prioritized finite logic program, and let $A \subseteq Lits$. Then, deciding whether $A \in \mathcal{PAS}(\mathcal{P})$ is polynomial (more precisely, complete for* P*).*

Indeed, for a full prioritized $\mathcal{P}$, it is easily seen that the test is polynomial. We have to check whether A is an answer set of P, which is well-known polynomial, and that $\lambda_{\mathcal{P}}(A) = A$, where $\lambda_{\mathcal{P}}(A) = C_{{}^{A}\mathcal{P}}(A)$. The dual reduct ${}^{A}\mathcal{P}$ is clearly computable in polynomial time, as is, given ${}^{A}\mathcal{P}$ and A, the set $C_{{}^{A}\mathcal{P}}(A)$. For an arbitrary $\mathcal{P}$, it suffices to consider a refinement of $<$ to some full prioritization $<'$ in which informally all rules that are not applied in A and possibly harm the reconstruction of A are ordered to the rear (discussed in detail in the full paper).

However, for the consistency problem, we have the following result.

Theorem 7.2 *Given a finite propositional prioritized program $\mathcal{P} = (P, <)$, deciding whether $\mathcal{PAS}(\mathcal{P}) \neq \emptyset$ is* NP-*complete. The problem is* NP-*hard even if $<$ is a total order and P does not involve strong negation.*

A literal L is a *consequence* of a prioritized program $\mathcal{P}$, denoted $\mathcal{P} \models L$, if L belongs to every preferred answer set of $\mathcal{P}$. The set of all consequences of $\mathcal{P}$ is denoted by $Cn(\mathcal{P})$.

Theorem 7.3 *Given a finite propositional prioritized program $\mathcal{P} = (P, <)$ and a literal L, deciding whether $\mathcal{P} \models L$ is* co-NP-*complete. The problem is* co-NP-*hard even if $<$ is a total order and P does not involve strong negation.*

As discussed in Section 6, for a fully prioritized program $\mathcal{P} = (P, <)$ the value $pvd_{\mathcal{P}}(A)$ amounts to the smallest number of switches of neighbored rules in $<$ such that A is a preferred answer set of the resulting program $\mathcal{P}' = (P, <')$; Moreover, we know from Proposition 6.4 that such a switching is always possible.

For the smallest number of switches, $|P|^2$ is an obvious upper bound. This bound can be tightened to

$$s = |P|-1+|P|-2+\ldots+1 = |P|(|P|-1)/2 = O(|P|^2),$$

which is the maximal number of switches performed by bubble sort when it sorts the input ordered by $<$ into $<'$.

The problem of computing $pvd_{\mathcal{P}}(A)$ can be rephrased as deciding whether $pvd_{\mathcal{P}}(A)$ is smaller than a given bound k. If a procedure for solving this problem is available, then we can actually compute $pvd_{\mathcal{P}}(A)$ in a binary search over $[0, s]$. As it turns out, deciding whether $pvd_{\mathcal{P}}(A) \leq k$ is a problem in NP. Thus, we can compute $pvd_{\mathcal{P}}(A)$ by making $O(\log |P|^2)$ many calls to an NP oracle.

The value of $pvd(\mathcal{P})$ can be computed similarly, if a subroutine solves the problem whether $pvd(\mathcal{P}) \leq k$ holds; also the latter problem is in NP. We thus obtain the following result.

Proposition 7.4 *Given a finite propositional prioritized program $\mathcal{P} = (P, <)$ and $A \in \mathcal{AS}(P)$, computing $pvd_{\mathcal{P}}(A)$ is in $\mathrm{F}\Delta_2^p[O(\log n)]$, i.e., possible in polynomial time with $O(\log n)$ many oracle queries, where n is the input size. Moreover, also computing $pvd(\mathcal{P})$ is in $\mathrm{F}\Delta_2^p[O(\log n)]$.*

We say that a literal L is a *weak consequence* of a prioritized program $\mathcal{P}$, denoted $\mathcal{P} \models_w L$, if L belongs to every weakly preferred answer set of $\mathcal{P}$. We denote by $Cn_w(\mathcal{P})$ the set of all weak consequences of $\mathcal{P}$. We have proven the following result.

Theorem 7.5 *Deciding, given a finite prioritized propositional program $\mathcal{P} = (P, <)$ and a literal L, whether $\mathcal{P} \models_w L$ is $\Delta_2^p[O(\log n)]$-complete. Hardness for $\Delta_2^p[O(\log n)]$ holds even if $<$ is a total order and P does not have strong negation.*

The inference problem $\mathcal{P} \models_w L$ can be solved as follows. First, compute $pvd(\mathcal{P})$ and then query an NP oracle whether some answer set A with $pvd_{\mathcal{P}}(A) \leq pvd(\mathcal{P})$ exists such that $L \notin A$. It can be seen that the oracle is in NP. Hence, a logarithmic number of NP oracle calls suffices, and the problem is in $\Delta_2^p[O(\log n)]$.

In the full paper, we also address the complexity of computing an arbitrary preferred (resp. weakly preferred) answer set.

8 Related Work and Conclusion

Apart from the approaches in the context of default logic which we have reviewed in Section 3, several approaches treating preferences in the context of logic programming have been described in the literature.

Kowalski and Sadri. In [20], Kowalski and Sadri have proposed to consider rules with negation in the head as exceptions to more general rules and to give them higher priority. The main achievement is that programs whose single answer set is inconsistent become consistent in the new semantics. The approach can hardly be viewed as a satisfactory treatment of preferences for several reasons: (1) Preferences are implicit and highly restricted; the asymmetric treatment of positive and negative information in this context seems unjustified from a knowledge representation perspective. (2) Fewer conclusions are obtained than in the original answer set semantics, contrary to what one would expect when preferences are taken into account.

It is, therefore, more reasonable to view Kowalski and Sadri's approach as a contribution to inconsistency handling rather than preference handling.

Pradhan and Minker. In [28] Pradhan and Minker show how priorities can be used to combine different potentially conflicting databases. Preference information is used to determine the information which has to be given up in the merging process.

There are two major differences between this and our work: (1) Pradhan and Minker consider Datalog databases only, that is, neither explicit negation nor negation as failure is admitted. Our approach is thus more general. (2) The underlying preference relation is a relation on atoms, not on rules as in our case. While this appears to be adequate for merging Datalog databases we strongly believe that the more fine grained distinctions which are possible in our approach are necessary for many knowledge representation problems.

Buccafurri et al. An approach that is closer in spirit to ours is ordered logic programming [7]. An ordered logic program is a set of components forming an inheritance hierarchy. Each component consists of a set of rules. Rules lower in the hierarchy have preference over those higher up in the hierarchy since the former are considered more specific. There are two main differences between ordered logic programs and our approach: (1) The preferences of ordered logic programs are predefined through the inheritance hierarchy. (2) Ordered logic programs use only one kind of negation, the distinction between negation as failure and classical negation is not expressible in the language.

Brewka. In the article [6], one of the authors of the present paper has described a prioritized version of the well-founded semantics for extended logic programs. In that paper, the preference information was expressed in the logical language and could thus be derived dynamically. Here we used an additional ordering relation outside the language. However, the representation in the language is not much more difficult and can be done along the lines of [5].

Zhang and Foo. In two subsequent papers [38, 37], Zhang and Foo have presented a framework for prioritized logic programs which at the syntactical level is very close to ours. Also in that framework, priorities are expressed by a strict partial ordering on the rules. However, the similarity is at the surface, since the approach to the semantics followed by Zhang and Foo is fundamentally different. Their semantics is operationally defined, in terms of a (nondeterministic) reduction of the initial prioritized program $\mathcal{P}$, which is reduced to an extended logic program $\mathcal{P}(P)$, whose answer sets are preferred answer sets of $\mathcal{P}$; the preferred answer sets of $\mathcal{P}$ are the answer sets of all programs P to which $\mathcal{P}$ may be (nondeterministically) reduced.

In general, Zhang and Foo's concept of preferred answer set is different from ours. Indeed, it is easy to see from the definitions that if the program without priorities has an answer set, then also the prioritized program has a preferred answer set. In this respect Zhang and Foo's notion is closer to our weakly preferred answer sets than to preferred answer sets. However, their concept does not coincide with weakly preferred answer sets either. Several examples illustrating the different behavior are contained in the full paper.

Gelfond and Son. In a recent paper, Gelfond and Son have also tackled the problem of adding priorities on defaults in the language of extended logic programs [17]. Their approach admits the specification of preference over rules in the object language. Rules are divided into definite rules and default (defeasible) rules. While definite rules must be strictly obeyed, default rules may be ignored if reasonable in a given context. The semantics of the language is defined in terms of a transformation of any program $\mathcal{P}$ into an extended logic program $t(\mathcal{P})$. The transformation is basically a meta-interpreter for programs.

Gelfond and Son point out that their approach is different from ours, and that it leads in general to different results. In particular, an answer set of a prioritized program may be generated which does not correspond to an answer set of the original, non-prioritized program. This cannot happen in our approach.

Delgrande and Schaub. In [9] Delgrande and Schaub have presented another approach for adding priorities to default theories. A set of default rules D, a background classical theory W, and a preference ordering $<$ (which is a strict partial ordering) on the default rules are transformed into a standard default theory $T = (D', W')$, by naming the defaults and introducing special purpose predicates.

Schaub and Delgrande's approach is different from ours. This can be illustrated by the default theory in Example 3.1 from Section 3.2 where Schaub and Delgrande select the extension $E_2 = Th(\{a, \neg b\})$, which shows that Principle I is violated.

Several issues remain for further work. One such issue is the extension to a syntactically and semantically richer framework,e.g., rules with disjunction in the heads [30, 15]. Another possible extension are constructs for expressing dynamic preferences in the object language. As we have briefly mentioned above, there are no principal obstacles to extending our approach in this direction.

Finally, we plan an experimental implementation of preferred and weakly preferred answer sets on top of the deductive reasoning system `dlv` [12].

Acknowledgements. The authors thank M. Gelfond, H. Herre and N. Leone for helpful and clarifying discussions on the topic of the paper. Furthermore, we are grateful to V. Marek for bibliographic hints on well-orderings, to H. Blair and A. Nerode for discussions on this subject.

References

[1] F. Baader and B. Hollunder. Priorities on Defaults with Prerequisite and their Application in Treating Specificity in Terminological Default Logic. *Journal of Automated Reasoning*, 15:41–68, 1995.

[2] C. Baral and M. Gelfond. Logic Programming and Knowledge Representation. *Journal of Logic Programming*, 19/20:73–148, 1994.

[3] S. Benferhat, C. Cayrol, D. Dubois, J. Lang, and H. Prade. Inconsistency Management and Prioritized Syntax-Based Entailment. In R. Bajcsy, editor, *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 640–645. Morgan Kaufman, 1993.

[4] N. Bidoit and C. Froidevaux. General Logic Databases and Programs: Default Semantics and Stratification. *Information and Computation*, 19:15–54, 1991.

[5] G. Brewka. Adding Priorities and Specificity to Default Logic. In *Proceedings JELIA '94*, LNAI 838, pages 247–260, 1994.

[6] G. Brewka. Well-Founded Semantics for Extended Logic Programs with Dynamic Preferences. *Journal of Artificial Intelligence Research*, pages 19–36, 1996.

[7] F. Buccafurri, N. Leone, and P. Rullo. Stable Models and their Computation for Logic Programming with Inheritance and True Negation. *Journal of Logic Programming*, 27(1):5–43, 1996.

[8] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and Expressive Power of Logic Programming. In *Proceedings of the Twelth IEEE International Conference on Computational Complexity (CCC '97)*, pages 82–101, 1997.

[9] J. Delgrande and T. Schaub. Compiling Reasoning With and About Preferences into Default Logic. In *Proceedings IJCAI '97*, pages 168–174, 1997.

[10] T. Eiter and G. Gottlob. On the Computational Cost of Disjunctive Logic Programming: Propositional Case. *Annals of Mathematics and Artificial Intelligence*, 15(3/4):289–323, 1995.

[11] T. Eiter and G. Gottlob. The Complexity of Logic-Based Abduction. *Journal of the ACM*, 42(1):3–42, January 1995.

[12] T. Eiter, N. Leone, C. Mateis, G. Pfeifer, and F. Scarcello. A Deductive System for Non-Monotonic Reasoning. In J. Dix, U. Furbach, and A. Nerode, editors, *Proceedings of the 4th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-97)*, number 1265 in LNCS, pages 364–375. Springer, 1997.

[13] M. Gelfond. Logic Programming and Reasoning with Incomplete Information. *Annals of Mathematics and Artificial Intelligence*, 12:89–116, 1994.

[14] M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In *Logic Programming: Proceedings Fifth Intl Conference and Symposium*, pages 1070–1080, Cambridge, Mass., 1988. MIT Press.

[15] M. Gelfond and V. Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing*, 9:365–385, 1991.

[16] M. Gelfond, H. Przymusinska, and T. Przymusinski. On the Relationship Between Circumscription and Negation as Failure. *Artificial Intelligence*, 38:75–94, 1989.

[17] M. Gelfond and R. Son. Reasoning with Prioritized Defaults. Manuscript, 1997.

[18] D. S. Johnson. A Catalog of Complexity Classes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A, chapter 2. Elsevier Science Publishers B.V. (North-Holland), 1990.

[19] K. Konolige. Hierarchic Autoepistemic Theories for Nonmonotonic Reasoning. In *Proceedings AAAI '88*, 1988.

[20] R. Kowalski and F. Sadri. Logic Programs with Exceptions. *New Generation Computing*, 9:387–400, 1991.

[21] J. Lloyd. *Foundations of Logic Programming*. Springer, Berlin, 1984, 1987.

[22] W. Marek and M. Truszczyński. Autoepistemic Logic. *Journal of the ACM*, 38(3):588–619, 1991.

[23] W. Marek and M. Truszczyński. Computing Intersection of Autoepistemic Expansions. In A. Nerode, W. Marek, and V. Subrahmanian, editors, *Proceedings of the 1st Intl. Workshop on Logic Programming and Nonmonotonic Reasoning (LPNMR-91)*, pages 37–50, Washington DC, July 1991. MIT Press.

[24] W. Marek and M. Truszczyński. *Nonmonotonic Logics – Context-Dependent Reasoning*. Springer, 1993.

[25] B. Nebel. How Hard is it to Revise a Belief Base? Technical Report TR No. 83, Inst. für Informatik, Albert-Ludwigs-Universität Freiburg, August 1996. To appear in: *Handbook on Defeasible Reasoning and Uncertainty Management Systems. Volume 2: Belief Change*, D. Dubois and H. Prade (eds).

[26] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[27] D. Poole. On the Comparison of Theories: Preferring the Most Specific Explanation. In *Proceedings IJCAI '85*, 1985.

[28] S. Pradhan and J. Minker. Combining Datalog Databases Using Priorities. *International Journal of Cooperative Intelligent Systems*, 1996.

[29] H. Prakken. *Logical Tools for Modelling Legal Argument*. Dissertation, Vrije Universiteit Amsterdam, 1993.

[30] T. Przymusinski. Stable Semantics for Disjunctive Programs. *New Generation Computing*, 9:401–424, 1991.

[31] R. Reiter. A Logic for Default Reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[32] J. Rintanen. On Specificity in Default Logic. In C. Mellish, editor, *Proceedings IJCAI '95*, pages 1474–1479. AAAI Press, 1995.

[33] J. Schlipf. Complexity and Undecidability Results in Logic Programming. *Annals of Mathematics and Artificial Intelligence*, 15(3/4):257–288, 1995.

[34] W. Sierpiński. *Cardinal and Ordinal Numbers*. Monografie Matematyczne tom. 34. Państwowe Wydawnictwo Naukowe, Warsaw, 1958.

[35] D. Touretzky. *The Mathematics of Inheritance*. Pitman Research Notes in Artificial Intelligence, London, 1986.

[36] D. Touretzky, R. Thomason, and J. Horty. A Skeptic's Menagerie: Conflictors, Preemptors, Reinstaters, and Zombies in Nonmonotonic Inheritance. In *Proceedings IJCAI '91*, 1991.

[37] Y. Zhang and N. Foo. Answer Sets for Prioritized Logic Programs. In *Proceedings ILPS 97*, pages 69–83, 1997.

[38] Y. Zhang and N. Foo. Towards Generalized Rule-Based Updates. In *Proceedings IJCAI '97*, pages 82–87, 1997.

Dynamic Logic Programming

J. J. Alferes
Dept. Matemática
Univ. Évora and
A.I. Centre
Univ. Nova de Lisboa,
2825 Monte da Caparica
Portugal

J. A. Leite, L. M. Pereira
A.I. Centre
Dept. Informática
Univ. Nova de Lisboa
2825 Monte da Caparica
Portugal

H. Przymusinska
Dept. Computer Science
California State
Polytechnic Univ.
Pomona, CA 91768
USA

T. C. Przymusinski
Dept. Computer Science
Univ. of California
Riverside, CA 92521
USA

Abstract

In this paper we investigate updates of knowledge bases represented by logic programs. In order to represent negative information, we use generalized logic programs which allow default negation not only in rule bodies but also in their heads.We start by introducing the notion of an update $P \oplus U$ of a logic program P by another logic program U. Subsequently, we provide a precise semantic characterization of $P \oplus U$, and study some basic properties of program updates. In particular, we show that our update programs generalize the notion of interpretation update.

We then extend this notion to compositional sequences of logic programs updates $P_1 \oplus P_2 \oplus \ldots$, defining a dynamic program update, and thereby introducing the paradigm of *dynamic logic programming.* This paradigm significantly facilitates modularization of logic programming, and thus modularization of non-monotonic reasoning as a whole.

Specifically, suppose that we are given a set of logic program modules, each describing a different state of our knowledge of the world. Different states may represent different time points or different sets of priorities or perhaps even different viewpoints. Consequently, program modules may contain mutually contradictory as well as overlapping information. The role of the dynamic program update is to employ the mutual relationships existing between different modules to precisely determine, at any given module composition stage, the declarative as well as the procedural semantics of the combined program resulting from the modules.

1 Introduction

Most of the work conducted so far in the field of logic programming has focused on representing *static* knowledge, i.e., knowledge that does not evolve with time. This is a serious drawback when dealing with *dynamic knowledge bases* in which not only the *extensional* part (the set of facts) changes dynamically but so does the *intensional* part (the set of rules).

In this paper we investigate updates of knowledge bases represented by logic programs. In order to represent negative information, we use *generalized logic programs* which allow default negation not only in rule bodies but also in their heads. This is needed, in particular, in order to specify that some atoms should became false, i.e., should be deleted. However, our updates are far more expressive than a mere insertion and deletion of facts. They can be specified by means of arbitrary program rules and thus they themselves are logic programs. Consequently, our approach demonstrates how to update one generalized logic program P (the initial program) by another generalized logic program U (the updating program), obtaining as a result a new, updated logic program $P \oplus U$.

Several authors have addressed the issue of updates of logic programs and deductive databases (see e.g. [9, 10, 1]), most of them following the so called *"interpretation update"* approach, originally proposed in [11, 5]. This approach is based on the idea of reducing the problem of finding an update of a knowledge base DB by another knowledge base U to the problem of finding updates of its individual interpretations (models[1]). More precisely, a knowledge base DB' is considered to be the update of a knowledge base DB by U if the set of models of DB'coincides with the set

[1]The notion of a model depends on the type of considered knowledge bases and on their semantics. In this paper we are considering (generalized) logic programs under the stable model semantics.

of updated models of DB, i.e. "the set of models of DB''' = "the set of updated models of DB". Thus, according to the interpretation update approach, the problem of finding an update of a *deductive* database DB is reduced to the problem of finding individual updates of all of its *relational instantiations* (models) M. Unfortunately such an approach suffers, in general, from several important drawbacks[2]:

- In order to obtain the update DB' of a knowledge base DB one has to first compute all the models M of DB (typically, a daunting task) and then individually compute their (possibly multiple) updates M_U by U. An update M_U of a given interpretation M is obtained by changing the status of only those literals in M that are *"forced"* to change by the update U, while keeping all the other literals intact by *inertia* (see e.g. [9, 10]).
- The updated knowledge base DB' is not defined directly but, instead, it is indirectly characterized as a knowledge base whose models coincide with the set of all updated models M_U of DB. In general, there is therefore no natural way of computing[3] DB' because the only straightforward candidate for DB' is the typically intractably large knowledge base DB'' consisting of all clauses that are entailed by all the updated models M_U of DB.
- Most importantly, while the *semantics* of the resulting knowledge base DB' indeed represents the *intended* meaning when just the *extensional* part of the knowledge base DB (the set of facts) is being updated, it leads to strongly *counter-intuitive* results when also the *intensional* part of the database (the set of rules) undergoes change, as the following example shows.

Example 1.1 *Consider the logic program P :*

$$P: \quad sleep \leftarrow not\, tv\_on \qquad tv\_on \leftarrow$$
$$watch\_tv \leftarrow tv\_on$$

whose $M = \{tv\_on, watch\_tv\}$ is its only stable model. Suppose now that the update U states that there is a power failure, and if there is a power failure then the TV is no longer on, as represented by the logic program U:

$$U: \quad not\, tv\_on \leftarrow power\_failure$$
$$power\_failure \leftarrow$$

According to the above mentioned interpretation approach to updating, we would obtain $M_U = \{power\_failure, watch\_tv\}$ as the only update of M by U. This is because $power\_failure$ needs to be added to the model and its addition forces us to make tv_on false. As a result, even though there is a power failure, we are still watching TV. However, by inspecting the initial program and the updating rules, we are likely to conclude that since "$watch\_tv$" was true only because "tv_on" was true, the removal of "tv_on" should make "$watch\_tv$" false by default. Moreover, one would expect "sleep" to become true as well. Consequently, the intended model of the update of P by U is the model $M_U^{'} = \{power\_failure, sleep\}$.

Suppose now that another update U_2 follows, described by the logic program:

$$U_2: \quad not\, power\_failure \leftarrow$$

stating that power is back up again. We should now expect the TV to be on again. Since power was restored, i.e. "$power\_failure$" is false, the rule "$not\, tv\_on \leftarrow power\_failure$" of U should have no effect and the truth value of "tv_on" should be obtained by inertia from the rule "$tv\_on \leftarrow$ " of the original program P.□

This example illustrates that, when updating knowledge bases, it is not sufficient to just consider the truth values of literals figuring in the heads of its rules because the truth value of their rule bodies may also be affected by the updates of other literals. In other words, it suggests that the *principle of inertia* should be applied not just to the individual literals in an interpretation but rather to entire *rules* of the knowledge base.

The above example also leads us to another important observation, namely, that the notion of an update DB' of one knowledge base DB by another knowledge base U should not just depend on the *semantics* of the knowledge bases DB and U, as it is the case with interpretation updates, but that it should also depend on their *syntax*. This is best illustrated by the following even simpler example:

Example 1.2 *Consider the logic program P :*

$$P: \quad innocent \leftarrow not\, found\_guilty$$

whose only stable model is $M = \{innocent\}$, because $found\_guilty$ is false by default. Suppose now that the

[2]In [1] the authors addressed the first two of the drawbacks mentioned below. They showed how to directly construct, given a logic program P, another logic program P' whose partial stable models are exactly the interpretation updates of the partial stable models of P. This eliminates both of these drawbacks (in the case when knowledge bases are logic programs) but it does not eliminate the third, most important drawback.

[3] In fact, in general such a database DB' may not exist at all.

update U states that the person has been found guilty:

$$U: \quad found\_guilty \leftarrow \; .$$

Using the interpretation approach, we would obtain $M_U = \{innocent, found\_guilty\}$ as the only update of M by U thus leading us to the counter-intuitive conclusion that the person is both innocent and guilty. This is because $found\_guilty$ must be added to the model M and yet its addition does not force us to make innocent false. However, it is intuitively clear that the interpretation $M'_U = \{found\_guilty\}$, stating that the person is guilty but no longer innocent, should be the only model of the updated program. Observe, however, that the program P is semantically equivalent *to the following program P' :*

$$P': \quad innocent \leftarrow$$

because the programs P and P' have exactly the same set of stable models, namely the model M. Nevertheless, while the model $M_U = \{innocent, found\_guilty\}$ is not the intended model of the update of P by U it is in fact the only reasonable model of the update of P' by U. □

In this paper we investigate the problem of updating knowledge bases represented by generalized logic programs and we propose a new solution to this problem that attempts to eliminate the drawbacks of the previously proposed approaches. Specifically, given one generalized logic program P (the so called initial program) and another logic program U (the updating program) we define a new generalized logic program $P \oplus U$ called the *update* of P by U. The definition of the updated program $P \oplus U$ does not require any computation of the models of either P or U and is in fact obtained by means of a simple, *linear-time* transformation of the programs P and U. As a result, the update transformation can be accomplished very efficiently and its *implementation* is quite straightforward[4].

Due to the fact that we apply the inertia principle not just to atoms but to entire program rules, the semantics of our updated program $P \oplus U$ avoids the drawbacks of interpretation updates and moreover it seems to properly represent the intended semantics. As mentioned above, the updated program $P \oplus U$ does not just depend on the *semantics* of the programs P and U, as it was the case with interpretation updates, but it also depends on their *syntax*. In order to make the meaning of the updated program clear and easily verifiable, we provide a *complete characterization* of the semantics of updated programs $P \oplus U$.

Nevertheless, while our notion of program update significantly differs from the notion of interpretation update, it coincides with the latter (as originally introduced in [9] under the name of *revision program* and later reformulated in the language of logic programs in [10]) when the initial program P is purely *extensional*, i.e., when the initial program is just a set of facts. Our definition also allows significant flexibility and can be easily modified to handle updates which incorporate *contradiction removal* or specify different inertia rules. Consequently, our approach can be viewed as introducing a general dynamic logic programming *framework* for updating programs which can be suitably modified to make it fit different application domains and requirements.

Finally, we extend the notion of program updates to sequences of programs, defining the so called *dynamic program updates*. The idea of dynamic updates is very simple and quite fundamental. Suppose that we are given a set of program modules P_s, indexed by different states of the world s. Each program P_s contains some knowledge that is supposed to be true at the state s. Different states may represent different time periods or different sets of priorities or perhaps even different viewpoints. Consequently, the individual program modules may contain mutually contradictory as well as overlapping information. The role of the dynamic program update $\bigoplus\{P_s : s \in S\}$ is to use the mutual relationships existing between different states (as specified by the order relation) to precisely determine, at any given state s, the *declarative* as well as the *procedural* semantics of the combined program, composed of all modules.

Consequently, the notion of a dynamic program update supports the important paradigm of *dynamic logic programming*. Given individual and largely *independent* program modules P_s describing our knowledge at different states of the world (for example, the knowledge acquired at different times), the dynamic program update $\bigoplus\{P_s : s \in S\}$ specifies the exact meaning of the union of these programs. Dynamic programming significantly facilitates *modularization* of logic programming and, thus, modularization of non-monotonic reasoning as a whole. Whereas traditional logic programming has concerned itself mostly with representing static knowledge, we show how to use logic programs to represent dynamically changing knowledge.

Our results extend and improve upon the approach initially proposed in [7], where the authors first argued that the principle of inertia should be applied to the rules of the initial program rather than to the individ-

[4]The implementation is available from: `http://www-ssdi.di.fct.unl.pt/~jja/updates/`.

ual literals in an interpretation. However, the specific update transformation presented in [7] suffered from some drawbacks and was not sufficiently general.

We begin in Section 2 by defining the language of *generalized logic programs,* which allow default negation in rule heads. We describe stable model semantics of such programs as a special case of the approach proposed earlier in [8]. In Section 3 we define the program update $P \oplus U$ of the initial program P by the updating program U. In Section 4 we provide a complete characterization of the semantics of program updates $P \oplus U$ and in Section 5 we study their basic properties. In Section 6 we introduce the notion of dynamic program updates. We close the paper with concluding remarks and notes on future research.

2 Generalized Logic Programs and their Stable Models

In order to represent *negative* information in logic programs and in their updates, we need more general logic programs that allow default negation $not\,A$ not only in premises of their clauses but also in their heads.[5]. We call such programs *generalized logic programs.* In this section we introduce generalized logic programs and extend the stable model semantics of normal logic programs [3] to this broader class of programs[6].

The class of generalized logic programs can be viewed as a special case of a yet broader class of programs introduced earlier in [8]. While our definition is different and seems to be simpler than the one used in [8], when restricted to the language that we are considering, the two definitions can be shown to be equivalent[7].

It will be convenient to *syntactically* represent generalized logic programs as *propositional Horn theories.* In particular, we will represent default negation $not\,A$ as a standard propositional variable (atom). Suppose that $\mathcal{K}$ is an arbitrary set of propositional variables whose names do not begin with a "*not*". By the propositional language $\mathcal{L}_{\mathcal{K}}$ *generated* by the set $\mathcal{K}$ we mean the language $\mathcal{L}$ whose set of propositional variables consists of:

$$\{A : A \in \mathcal{K}\} \cup \{not\,A : A \in \mathcal{K}\}.$$

[5]For further motivation and intuitive reading of logic programs with default negations in the heads see [8].

[6]In a forthcoming paper we extend our results to 3-valued (partial) models of logic programs, and, in particular, to well-founded models.

[7]Note that the class of generalized logic programs differs from the class of programs with the so called *"classical"* negation [4] which allow the use of *strong* rather than default negation in their heads.

Atoms $A \in \mathcal{K}$, are called *objective atoms* while the atoms $not\,A$ are called *default atoms.* From the definition it follows that the two sets are disjoint.

By a *generalized logic program* P in the language $\mathcal{L}_{\mathcal{K}}$ we mean a finite or infinite set of propositional Horn clauses of the form:

$$L \leftarrow L_1, \ldots, L_n$$

where L and L_i are atoms from $\mathcal{L}_{\mathcal{K}}$. If all the atoms L appearing in heads of clauses of P are objective atoms, then we say that the logic program P is *normal.* Consequently, from a syntactic standpoint, a logic program is simply viewed as a propositional Horn theory. However, its *semantics* significantly differs from the semantics of classical propositional theories and is determined by the class of stable models defined below.

By a (2-valued) *interpretation* M of $\mathcal{L}_{\mathcal{K}}$ we mean any set of atoms from $\mathcal{L}_{\mathcal{K}}$ that satisfies the condition that for any A in $\mathcal{K}$, precisely one of the atoms A or $not\,A$ belongs to M. Given an interpretation M we define:

$$\begin{aligned} M^+ &= \{A \in \mathcal{K} : A \in M\} \\ M^- &= \{not\,A : not\,A \in M\} = \\ &= \{\,not\,A : A \notin M\}. \end{aligned}$$

Definition 2.1 (Stable models of generalized logic programs) *We say that a (2-valued) interpretation M of $\mathcal{L}_{\mathcal{K}}$ is a stable model of a generalized logic program P if M is the least model of the Horn theory $P \cup M^-$:*

$$M = Least(P \cup M^-),$$

or, equivalently, if $M = \{L : L$ is an atom and $P \cup M^- \vdash L\}$. □

Example 2.1 *Consider the program:*

$$\begin{array}{ccc} a \leftarrow not\,b & c \leftarrow b & e \leftarrow not\,d \\ not\,d \leftarrow not\,c, a & d \leftarrow not\,e & \end{array}$$

and let $\mathcal{K} = \{a, b, c, d, e\}$. This program has precisely one stable model $M = \{a, e, not\,b, not\,c, not\,d\}$. To see that M is stable we simply observe that:

$$M = Least(P \cup \{not\,b, not\,c, not\,d\}).$$

The interpretation $N = \{not\,a, not\,e, b, c, d\}$ is not a stable model because:

$$N \neq Least(P \cup \{not\,e, not\,a\}). \;\square$$

Following an established tradition, from now on we will be omitting the default (negative) atoms when describing interpretations and models. Thus the above model M will be simply listed as $M = \{a, e\}$. The following Proposition easily follows from the definition of stable models.

Proposition 2.1 *An interpretation M of $\mathcal{L}_{\mathcal{K}}$ is a stable model of a generalized logic program P if and only if*

$$M^+ = \{A : A \in \mathcal{K} \text{ and } \frac{P}{M} \vdash A\}$$

and

$$M^- \supseteq \{not\, A : A \in \mathcal{K} \text{ and } \frac{P}{M} \vdash not A\},$$

where $\frac{P}{M}$ denotes the Gelfond-Lifschitz transform [3] of P w.r.t. M. □

Clearly, the second condition in the above Proposition is always vacuously satisfied for normal programs and therefore we immediately obtain:

Proposition 2.2 *The class of stable models of generalized logic programs extends the class of stable models of normal programs [3].* □

3 Program Updates

Suppose that $\mathcal{K}$ is an arbitrary set of propositional variables, and P and U are two generalized logic programs in the language $\mathcal{L} = \mathcal{L}_{\mathcal{K}}$. By $\widehat{\mathcal{K}}$ we denote the following superset of $\mathcal{K}$:

$$\widehat{\mathcal{K}} = \mathcal{K} \cup \{A^-,\ A_P,\ A_P^-,\ A_U,\ A_U^- : A \in \mathcal{K}\}.$$

This definition assumes that the original set $\mathcal{K}$ of propositional variables does not contain any of the newly added symbols of the form $A^-, A_P, A_P^-, A_U, A_U^-$ so that they are all disjoint sets of symbols. If $\mathcal{K}$ contains any such symbols then they have to be *renamed* before the extension of $\mathcal{K}$ takes place. We denote by $\widehat{\mathcal{L}} = \mathcal{L}_{\widehat{\mathcal{K}}}$ the extension of the language $\mathcal{L} = \mathcal{L}_{\mathcal{K}}$ generated by $\widehat{\mathcal{K}}$.

Definition 3.1 (Program Updates) *Let P and U be generalized programs in the language $\mathcal{L}$. We call P the original program and U the updating program. By the update of P by U we mean the generalized logic program $P \oplus U$, which consists of the following clauses in the extended language $\widehat{\mathcal{L}}$:*

(RP) Rewritten original program clauses:

$$\begin{array}{lcll} A_P & \leftarrow & B_1, \ldots, B_m, C_1^-, \ldots, C_n^- & (1) \\ A_P^- & \leftarrow & B_1, \ldots, B_m, C_1^-, \ldots, C_n^- & (2) \end{array}$$

for any clause:

$$A \leftarrow B_1,\ \ldots,\ B_m,\ not\, C_1,\ \ldots,\ not\, C_n$$

and

$$not\, A \leftarrow B_1,\ \ldots,\ B_m, not\, C_1,\ \ldots,\ not\, C_n$$

respectively, in the original program P. The rewritten clauses are obtained from the original ones by replacing atoms A (respectively, the atoms $not\, A$) occurring in their heads by the atoms A_P (respectively, A_P^-) and by replacing negative premises $not\, C$ by C^-.

(RU) Rewritten updating program clauses:

$$\begin{array}{lcll} A_U & \leftarrow & B_1, \ldots, B_m, C_1^-, \ldots, C_n^- & (3) \\ A_U^- & \leftarrow & B_1, \ldots, B_m, C_1^-, \ldots, C_n^- & (4) \end{array}$$

for any clause:

$$A \leftarrow B_1,\ \ldots,\ B_m,\ not\, C_1,\ \ldots,\ not\, C_n$$

and, respectively,

$$not\, A \leftarrow B_1,\ \ldots,\ B_m, not\, C_1,\ \ldots,\ not\, C_n$$

in the updating program U. The rewritten clauses are obtained from the original ones by replacing atoms A (respectively, the atoms $not\, A$) occurring in their heads by the atoms A_U (respectively, A_U^-) and by replacing negative premises $not\, C$ by C^-.

(UR) Update rules:

$$A \leftarrow A_U \qquad A^- \leftarrow A_U^- \qquad (5)$$

for all objective atoms $A \in \mathcal{K}$. The update rules state that an atom A must be true (respectively, false) in $P \oplus U$ if it is true (respectively, false) in the updating program U.

(IR) Inheritance rules:

$$A \leftarrow A_P, not\, A_U^- \qquad A^- \leftarrow A_P^-, not\, A_U \qquad (6)$$

for all objective atoms $A \in \mathcal{K}$. The inheritance rules say that an atom A (respectively, A^-) in $P \oplus U$ is inherited (by inertia) from the original program P provided it is not rejected (i.e., forced to be false) by the updating program U. More precisely, an atom A is true (respectively, false) in $P \oplus U$ if it is true (respectively, false) in the original program P, provided it is not made false (respectively, true) by the updating program U.

(DR) Default rules:

$$A^- \leftarrow not\, A_P, not\, A_U \qquad not\, A \leftarrow A^- \qquad (7)$$

for all objective atoms $A \in \mathcal{K}$. The first default rule states that an atom A in $P \oplus U$ is false if it is neither true in the original program P nor in the updating program U. The second says that if an atom is false then it can be assumed to be false by default. It ensures that A and A^- cannot both be true. □

It is easy to show that any model N of $P \oplus U$ is *coherent*, i.e., A is true (respectively, false) in N iff A^- is false (respectively, true) in N, for any $A \in \mathcal{K}$. In other words, every stable model of $P \oplus U$ satisfies the constraint $not\, A \equiv A^-$. Consequently, A^- can be simply regarded as an internal (meta-level) representation of the default negation $not\, A$ of A.

Example 3.1 *Consider the programs P and U from Example 1.1:*

$$\begin{array}{lll} P: & sleep \leftarrow not\, tv\_on & tv\_on \leftarrow \\ & watch\_tv \leftarrow tv\_on & \end{array}$$

$$\begin{array}{ll} U: & not\ tv\_on \leftarrow power\_failure \\ & power\_failure \leftarrow \end{array}$$

The update of the program P by the program U is the logic program $P \oplus U = (RP) \cup (RU) \cup (UR) \cup (IR) \cup (DR)$, where:

$$\begin{array}{lll} RP: & sleep_P \leftarrow tv\_on^- & tv\_on_P \leftarrow \\ & watch\_tv_P \leftarrow tv\_on & \end{array}$$

$$\begin{array}{ll} RU: & tv\_on_U^- \leftarrow power\_failure \\ & power\_failure_U \leftarrow \end{array}$$

It is easy to verify that $M = \{power\_failure, sleep\}$ is the only stable model (modulo irrelevant literals) of $P \oplus U$. □

4 Semantic Characterization of Program Updates

In this section we provide a complete semantic characterization of update programs $P \oplus U$ by describing their stable models. This characterization shows precisely how the semantics of the update program $P \oplus U$ depends on the syntax and semantics of the programs P and U.

Let P and U be *fixed* generalized logic programs in the language $\mathcal{L}$. Since the update program $P \oplus U$ is defined in the extended language $\widehat{\mathcal{L}}$, we begin by showing how interpretations of the language $\mathcal{L}$ can be extended to interpretations of the extended language $\widehat{\mathcal{L}}$.

Definition 4.1 (Extended Interpretation) *For any interpretation M of $\mathcal{L}$ we denote by $\widehat{M}$ its extension to an interpretation of the extended language $\widehat{\mathcal{L}}$ defined, for any atom $A \in \mathcal{K}$, by the following rules:*

$$\begin{array}{lll} A^- \in \widehat{M} & \textit{iff} & not\, A \in M \\ A_P \in \widehat{M} & \textit{iff} & \exists\ A \leftarrow Body \in P \ \ and \ \ M \models Body \\ A_P^- \in \widehat{M} & \textit{iff} & \exists\ not\, A \leftarrow Body \in P \\ & & \quad and \ \ M \models Body \\ & & \\ A_U \in \widehat{M} & \textit{iff} & \exists A \leftarrow Body \in U \ and \ \ M \models Body \\ A_U^- \in \widehat{M} & \textit{iff} & \exists\ not\, A \leftarrow Body \in U \\ & & \quad and \ \ M \models Body. \end{array}$$ □

We will also need the following definition:

Definition 4.2 *For any model M of the program U in the language $\mathcal{L}$ define:*

$$Defaults[M] = \{not\ A : M \models \neg Body, \forall (A \leftarrow Body) \in P \cup U\};$$

$$\begin{array}{l} Rejected[M] = \\ \{A \leftarrow Body \in P : \exists\ (not\ A \leftarrow Body' \in U) \\ \quad and \ \ M \models Body'\} \\ \cup \\ \{not\ A \leftarrow Body \in P : \exists\ (A \leftarrow Body' \in U) \\ \quad and \ \ M \models Body'\}; \end{array}$$

$$Residue[M] = P \cup U - Rejected[M].$$ □

The set $Defaults[M]$ contains default negations $not\, A$ of all *unsupported* atoms A, i.e., atoms that have the property that the body of every clause from $P \cup U$ with the head A is false in M. Consequently, negation $not\, A$ of these unsupported atoms A can be assumed by default. The set $Rejected[M] \subseteq P$ represents the set of clauses of the original program P that are *rejected* (or contradicted) by the update program U and its model M. The residue $Residue[M]$ consists of all clauses in the union $P \cup U$ of programs P and U that were *not* rejected by the update program U. Note that all the three sets depend on the model M as well as on the *syntax* of the programs P and U.

Now we are able to describe the semantics of the update program $P \oplus U$ by providing a complete characterization of its stable models.

Theorem 4.1 (Characterization of stable models of update programs) *An interpretation N of the language $\widehat{\mathcal{L}} = \mathcal{L}_{\widehat{\mathcal{K}}}$ is a stable model of the update $P \oplus U$ if and only if N is the extension $N = \widehat{M}$ of a model M of U that satisfies the condition:*

$$M = Least(P \cup U - Rejected[M] \cup Defaults[M]),$$

or $M = Least(Residue[M] \cup Defaults[M])$, equivalently. □

Example 4.1 *Consider again the programs P and U from Example 1.1. Let $M = \{power\_failure, sleep\}$. We obtain:*

$$\begin{aligned} Defaults[M] &= \{not\, watch\_tv,\} \\ Rejected[M] &= \{tv\_on \leftarrow\} \\ Residue[M] &= \left\{ \begin{array}{l} sleep \leftarrow not\, tv\_on \\ watch\_tv \leftarrow tv\_on \\ not\, tv\_on \leftarrow power\_failure \\ power\_failure \leftarrow \end{array} \right\} \end{aligned}$$

and thus it is easy to see that

$$M = Least(Residue[M] \cup Defaults[M]).$$

Consequently, $\widehat{M}$ is a stable model of the update program $P \oplus U$. □

5 Properties of Program Updates

In this section we study the basic properties of program updates. Since $Defaults[M] \subseteq M^-$, we conclude that the condition $M = Least(Residue[M] \cup Defaults[M])$ clearly implies $M = Least(Residue[M] \cup M^-)$ and thus we immediately obtain:

Proposition 5.1 *If N is a stable model of $P \oplus U$ then its restriction $M = N|\mathcal{L}$ to the language $\mathcal{L}$ is a stable model of $Residue[M]$.* □

However, the condition $M = Least(Residue[M] \cup Defaults[M])$ says much more than just that M is a stable model of $Residue[M]$. It says that M is completely *determined* by the set $Defaults[M]$, i.e., by the set of negations of unsupported atoms that can be assumed false by default.

Clearly, if M is a stable model of $P \cup U$ then $Rejected[M] = \emptyset$ and $Defaults[M] = M^-$, which implies:

Proposition 5.2 *If M is a stable model of the union $P \cup U$ of programs P and U then its extension $N = \widehat{M}$ is a stable model of the update program $P \oplus U$. Thus, the semantics of the update program $P \oplus U$ is always weaker than or equal to the semantics of the union $P \cup U$ of programs P and U.* □

In general, the converse of the above result does not hold. In particular, the union $P \cup U$ may be a contradictory program with no stable models.

Example 5.1 *Consider again the programs P and U from Example 1.1. It is easy to see that $P \cup U$ is contradictory.* □

If either P or U is empty and M is a stable model of $P \cup U$ then $Rejected[M] = \emptyset$ and therefore M is also a stable model of $P \oplus U$.

Proposition 5.3 *If either P or U is empty then M is a stable model of $P \cup U$ iff $N = \widehat{M}$ is a stable model of $P \oplus U$. Thus, in this case, the semantics of the update program $P \oplus U$ coincides with the semantics of the union $P \cup U$.* □

Proposition 5.4 *If both P and U are normal programs (or if both have only clauses with default atoms not A in their heads) then M is a stable model of $P \cup U$ iff $N = \widehat{M}$ is a stable model of $P \oplus U$. Thus, in this case the semantics of the update program $P \oplus U$ also coincides with the semantics of the union $P \cup U$ of programs P and U.* □

5.1 Program Updates Generalize Interpretation Updates

In this section we show that *interpretation updates*, originally introduced under the name *"revision programs"* by Marek and Truszczynski [9], and subsequently given a simpler characterization by Przymusinski and Turner [10], constitute a special case of program updates. Here, we identify the *"revision rules"*:

$$\begin{aligned} in(A) &\leftarrow in(B), out(C) \\ out(A) &\leftarrow in(B), out(C) \end{aligned}$$

used in [9], with the following generalized logic program clauses:

$$\begin{aligned} A &\leftarrow B, not\, C \\ not\, A &\leftarrow B, not\, C. \end{aligned}$$

Theorem 5.1 (Program updates generalize interpretation updates) *Let I be any interpretation and U any updating program in the language $\mathcal{L}$. Denote by P_I the generalized logic program in $\mathcal{L}$ defined by*

$$P_I = \{A \leftarrow : A \in I\} \cup \{not\, A \leftarrow : not\, A \in I\}.$$

Then $\widehat{J}$ is a stable model of the program update $P_I \oplus U$ of the program P_I by the program U iff J is an interpretation update of I by U (in the sense of [9]). □

This theorem shows that when the initial program P is purely *extensional*, i.e., contains only positive or negative *facts*, then the interpretation update of P by U is semantically equivalent to the updated program $P \oplus U$. As shown by the Examples 1.1 and 1.2, when P contains deductive rules then the two notions become significantly different.

Remark 5.1 *It is easy to see that, optionally, we could include only positive facts $A \leftarrow$ in the program P_I thus making it a normal program.* □

5.2 Adding Strong Negation

We now show that it is easy to add *strong negation* $-A$ ([4],[2]) to generalized logic programs. This demonstrates that the class of generalized logic programs is at least as expressive as the class of logic programs with strong negation. It also allows us to update logic programs with strong negation and to use strong negation in updating programs.

Definition 5.1 (Adding strong negation) *Let $\mathcal{K}$ be an arbitrary set of propositional variables. In order to add strong negation to the language $\mathcal{L} = \mathcal{L}_{\mathcal{K}}$ we just augment the set $\mathcal{K}$ with new propositional symbols $\{-A : A \in \mathcal{K}\}$, obtaining the new set $\mathcal{K}^*$, and consider the extended language $\mathcal{L}^* = \mathcal{L}_{\mathcal{K}^*}$. In order to ensure that A and $-A$ cannot be both true we also assume, for all $A \in \mathcal{K}$, the following strong negation axioms, which themselves are generalized logic program clauses:*

$$\begin{array}{ll} (SN1) & \mathit{not}\, A \leftarrow\ -A \\ (SN2) & \mathit{not}\, -A \leftarrow A. \end{array}$$

Remark 5.2 *In order to prevent the strong negation rules (SN) from being inadvertently overruled by the updating program U, one may want to make them always part of the most current updating program (see the next section).* □

6 Dynamic Program Updates

In this section we introduce the notion of *dynamic program update* $\bigoplus\{P_s : s \in S\}$ over an ordered set $\mathcal{P} = \{P_s : s \in S\}$ of logic programs which provides an important generalization of the notion of single program updates $P \oplus U$ introduced in Section 3.

The idea of dynamic updates, inspired by [6], is simple and quite fundamental. Suppose that we are given a set of program modules P_s, indexed by different states of the world s. Each program P_s contains some knowledge that is supposed to be true at the state s. Different states may represent different time periods or different sets of priorities or perhaps even different viewpoints. Consequently, the individual program modules may contain mutually contradictory as well as overlapping information. The role of the dynamic program update $\bigoplus\{P_s : s \in S\}$ is to use the mutual relationships existing between different states (and specified in the form of the ordering relation) to precisely determine, at any given state s, the *declarative* as well as the *procedural* semantics of the combined program, composed of all modules.

Consequently, the notion of a dynamic program update supports the important paradigm of *dynamic logic programming.* Given individual and largely *independent* program modules P_s describing our knowledge at different states of the world (for example, the knowledge acquired at different times), the dynamic program update $\bigoplus\{P_s : s \in S\}$ specifies the exact meaning of the union of these programs. Dynamic programming significantly facilitates modularization of logic programming and, thus, modularization of non-monotonic reasoning as a whole.

Suppose that $\mathcal{P} = \{P_s : s \in S\}$ is a finite or infinite sequence of generalized logic programs in the language $\mathcal{L} = \mathcal{L}_{\mathcal{K}}$, indexed by the set $S = \{1, 2, \ldots, n, \ldots\}$. We will call elements s of the set $S \cup \{0\}$ *states* and we will refer to 0 as the *initial state.* If S has the *largest* element then we will denote it by max.

Remark 6.1 *Instead of a linear sequence of states $S \cup \{0\}$ one could as well consider any finite or infinite ordered set with the smallest element s_0 and with the property that every state s other than s_0 has an immediate predecessor $s-1$ and that $s_0 = s - n$, for some finite n. In particular, one may use a finite or infinite tree with the root s_0 and the property that every node (state) has only a finite number of ancestors.* □

By $\overline{\mathcal{K}}$ we denote the following superset of the set $\mathcal{K}$ of propositional variables:

$$\begin{array}{rl} \overline{\mathcal{K}} = \mathcal{K} \cup \{ & A^-, A_s,\ A_s^-, A_{P_s},\ A_{P_s}^-, \mathit{reject}(A_s), \\ & \mathit{reject}(A_s^-) : A \in \mathcal{K},\ s \in S \cup \{0\}\}. \end{array}$$

As before, this definition assumes that the original set $\mathcal{K}$ of propositional variables does not contain any of the newly added symbols of the form $A^-, A_s,\ A_s^-, A_{P_s},\ A_{P_s}^-, \mathit{reject}(A_s), \mathit{reject}(A_s^-)$ so that they are all disjoint sets of symbols. If the original language $\mathcal{K}$ contains any such symbols then they have to be *renamed* before the extension of $\mathcal{K}$ takes place.

We denote by $\overline{\mathcal{L}} = \mathcal{L}_{\overline{\mathcal{K}}}$ the extension of the language $\mathcal{L} = \mathcal{L}_{\mathcal{K}}$ generated by $\overline{\mathcal{K}}$.

Definition 6.1 (Dynamic Program Update) *By the dynamic program update over the sequence of updating programs $\mathcal{P} = \{P_s : s \in S\}$ we mean the logic program $\uplus\mathcal{P}$, which consists of the following clauses in the extended language $\overline{\mathcal{L}}$:*

(RP) Rewritten program clauses:

$$A_{P_s} \leftarrow B_1, \ldots, B_m, C_1^-, \ldots, C_n^- \quad (8)$$
$$A_{P_s}^- \leftarrow B_1, \ldots, B_m, C_1^-, \ldots, C_n^- \quad (9)$$

for any clause:

$$A \leftarrow B_1, \ \ldots, \ B_m, \ not\, C_1, \ \ldots, \ not\, C_n$$

respectively, for any clause:

$$not\, A \leftarrow B_1, \ \ldots, \ B_m, not\, C_1, \ \ldots, \ not\, C_n$$

in the program P_s, where $s \in S$. The rewritten clauses are simply obtained from the original ones by replacing atoms A (respectively, the atoms $not\, A$) occurring in their heads by the atoms A_{P_s} (respectively, $A_{P_s}^-$) and by replacing negative premises $not\, C$ by C^-.

(UR) Update rules:

$$A_s \leftarrow A_{P_s}; \qquad A_s^- \leftarrow A_{P_s}^- \quad (10)$$

for all objective atoms $A \in \mathcal{K}$ and for all $s \in S$. The update rules state that an atom A must be true (respectively, false) in the state $s \in S$ if it is true (respectively, false) in the updating program P_s.

(IR) Inheritance rules:

$$A_s \leftarrow A_{s-1}, not\, reject(A_{s-1});$$
$$A_s^- \leftarrow A_{s-1}^-, not\, reject(A_{s-1}^-) \quad (11)$$

$$reject(A_{s-1}) \leftarrow A_{P_s}^-; \qquad reject(A_{s-1}^-) \leftarrow A_{P_s} \quad (12)$$

for all objective atoms $A \in \mathcal{K}$ and for all $s \in S$. The inheritance rules say that an atom A is true (respectively, false) in the state $s \in S$ if it is true (respectively, false) in the previous state $s-1$ and it is not rejected, *i.e., forced to be false (respectively, true), by the updating program P_s. The addition of the special predicate* reject, *although not strictly needed at this point, allows us to impose later on additional restrictions on the inheritance by inertia (see Section 6.2).*

(DR) Default rules (describing the initial state):

$$A_0^-, \quad (13)$$

for all objective atoms $A \in \mathcal{K}$. Default rules describe the initial state 0 by making all objective atoms initially false. □

Observe that the dynamic program update $\uplus\mathcal{P}$ is a normal logic program, i.e., it does not contain default negation in heads of its clauses. Moreover, only the inheritance rules contain default negation in their bodies. Also note that the program $\uplus\mathcal{P}$ does not contain the atoms A or A^-, where $A \in \mathcal{K}$, in heads of its clauses. These atoms appear only in the bodies of rewritten program clauses. The notion of the dynamic program update $\bigoplus_s\mathcal{P}$ at a given state $s \in S$ changes that.

Definition 6.2 (Dynamic Program Update at a Given State) *Given a fixed state $s \in S$, by the dynamic program update at the state s, denoted by $\bigoplus_s\mathcal{P}$, we mean the dynamic program update $\uplus\mathcal{P}$ augmented with the following:*

Current State Rules CS(s):

$$A \leftarrow A_s \qquad A^- \leftarrow A_s^- \qquad not\, A \leftarrow A_s^- \quad (14)$$

for all objective atoms $A \in \mathcal{K}$. Current state rules specify the current state s in which the updated program is being evaluated and determine the values of the atoms A, A^- and $not\, A$. In particular, if the set S has the largest element max *then we simply write $\bigoplus\mathcal{P}$ instead of $\bigoplus_{max}\mathcal{P}$.* □

Mark that whereas for any state s $\uplus\mathcal{P}$ is not required to be coherent, $\bigoplus_s\mathcal{P}$ must be so.

The notion of a dynamic program update generalizes the previously introduced notion of an update $P \oplus U$ of two programs P and U.

Theorem 6.1 *Let P_1 and P_2 be arbitrary generalized logic programs and let $S = \{1, 2\}$. The dynamic program update $\bigoplus\{P_1, P_2\} = \bigoplus_2\{P_1, P_2\}$ at the state $max = 2$ is semantically equivalent to the program update $P_1 \oplus P_2$ defined in Section 3.* □

6.1 Examples

Example 6.1 *Let $\mathcal{P} = \{P_1, P_2, P_3\}$, where P_1, P_2 and P_3 are as follows:*

$$\begin{array}{ll} P_1: & sleep \leftarrow not\, tv\_on \\ & watch\_tv \leftarrow tv\_on \\ & tv\_on \leftarrow \end{array}$$

$$\begin{array}{ll} P_2: & not\, tv\_on \leftarrow power\_failure \\ & power\_failure \leftarrow \end{array}$$

$$\begin{array}{ll} P_3: & not\, power\_failure \leftarrow \end{array}$$

The dynamic program update over $\mathcal{P}$ is the logic program $\biguplus\mathcal{P} = (RP_1) \cup (RP_2) \cup (RP_3) \cup (UR) \cup (IR) \cup (DR)$, where

$$\begin{array}{ll} RP_1: & sleep_{P_1} \leftarrow tv\_on^- \\ & watch\_tv_{P_1} \leftarrow tv\_on \\ & tv\_on_{P_1} \leftarrow \end{array}$$

$$\begin{array}{ll} RP_2: & tv\_on^-_{P_2} \leftarrow power\_failure \\ & power\_failure_{P_2} \leftarrow \end{array}$$

$$\begin{array}{ll} RP_3: & power\_failure^-_{P_3} \leftarrow \end{array}$$

and the dynamic program update at the state s is $\bigoplus_s \mathcal{P} = \biguplus\mathcal{P} \cup CS(s)$. Consequently, as intended, $\bigoplus_1 \mathcal{P}$ has a single stable model $M_1 = \{tv\_on, watch\_tv\}$; $\bigoplus_2 \mathcal{P}$ has a single stable model $M_2 = \{sleep, power\_failure\}$ and $\bigoplus \mathcal{P} = \bigoplus_3 \mathcal{P}$ has a single stable model $M_3 = \{tv\_on, watch\_tv\}$ (all models modulo irrelevant literals). Moreover. $\bigoplus_2 \mathcal{P}$ is semantically equivalent to $P_1 \oplus P_2$. □

As mentioned in the Introduction, in dynamic logic programming, logic program modules describe states of our knowledge of the world, where different states may represent different time points or different sets of priorities or even different viewpoints. It is not our purpose in this paper to discuss in detail how to apply dynamic logic programming to any of these application domains[8]. However, since all of the examples presented so far relate different program modules with changing time, below we illustrate how to use dynamic logic programming to represent the well known problem in the domain of taxonomies by using priorities among rules.

Example 6.2 *Consider the well-known problem of flying birds. In this example we have several rules with different priorities. First, the animals-do-not-fly rule, which has the lowest priority; then the birds-fly rule with a higher priority; the penguins-do-not-fly rule with an even higher priority; and, finally, with the highest priority, all the rules describing the actual taxonomy (penguins are birds, birds are animal, etc). This can be coded quite naturally in dynamic logic programming:*

$$\begin{array}{ll} P_1: & not\, fly(X) \leftarrow animal(X) \\ P_2: & fly(X) \leftarrow bird(X) \\ P_3: & not\, fly(X) \leftarrow penguin(X) \\ P_4: & animal(X) \leftarrow bird(X) \\ & bird(X) \leftarrow penguin(X) \\ & animal(pluto) \\ & bird(duffy) \\ & penguin(tweety) \end{array}$$

The reader can check that, as intended, the dynamic logic program at state 4, i.e. $\bigoplus_4\{P_1, P_2, P_3, P_4\}$, has a single stable model where $fly(duffy)$ is true, and both $fly(pluto)$ and $fly(tweety)$ are false. □

Sometimes it is useful to have some kind of a background knowledge, i.e., knowledge that is true in every program module or state. This is true, for example, in the case of strong negation axioms presented in Section 5.2, because these axioms must be true in every program module. This is also true in the case of laws in the domain of actions and effects of action. These laws must be valid in every state and at any time (for example, the law saying that if there is no power then the tv must be off).

Rules describing background knowledge, i.e., background rules, are easily representable in dynamic logic programming: if a rule is valid in every program state, simply add that rule to every program state. However, this is not a very practical, and, especially, not a very efficient way of representing background rules. Fortunately, in dynamic program updates at a given state s, adding a rule to every state is equivalent to adding that rule only in the state s:

Proposition 6.1 *Let $\bigoplus_s \mathcal{P}$ be a dynamic program update at state s, and let r be a rule such that $\forall P_i \in \mathcal{P}, r \in P_i$. Let $\mathcal{P}'$ be the set of logic program obtained from $\mathcal{P}$ such that $P_s \in \mathcal{P}'$ and*

$$\forall i \neq s, P'_i = P_i - \{r\} \in \mathcal{P}' \text{ iff } P_i \in \mathcal{P}$$

There is a one-to-one correspondence between the stable models of $\bigoplus_s \mathcal{P}$ restricted to $\mathcal{K}$ and the stable models of $\bigoplus_s \mathcal{P}'$ restricted to $\mathcal{K}$. □

Thus, such background rules need not necessarily be added to every program state. Instead, they can sim-

[8]In fact, this is the subject of our ongoing research. In particular, the application of dynamic logic programming to the domain of actions is the subject of a forthcoming paper.

ply be added at the state s. Such background rules are therefore similar to the axioms $CS(s)$, which are added only when the state s is fixed. In particular, considering the background rules in every program state is equivalent to considering them *as part of* the axioms $CS(s)$.

A more detailed discussion of the formalization and usage of background knowledge will appear in the aforementioned forthcoming paper on the application of dynamic logic programming to the domain of actions.

6.2 Limiting the Inheritance by Inertia

Inheritance rules (IR) describe *the rules of inertia*, i.e., the rules guiding the inheritance of knowledge from one state s to the next state s'. In particular, they prevent the inheritance of knowledge that is explicitly contradicted in the new state s'. However, inheritance can be limited even further, by means of specifying additional rules for the predicate *reject*.

One important example of such additional constraints imposed on the inertia rules involves filtering out from the current state s' of any incoherence (inconsistency, contradiction) that occurred in the previous state s. Such inconsistency could have already existed in the previous state s or could have been caused by the new information added at the current state s'. In order to eliminate such contradictory information, it suffices to add to the definition of *reject* the following two rules:

$$reject(A_{s-1}) \leftarrow A^{-}_{s-1} \qquad reject(A^{-}_{s-1}) \leftarrow A_{s-1}$$

Similarly, the removal of contradictions brought about by the strong negation axioms of 5.1 can be achieved by adding the rules:

$$reject(A_{s-1}) \leftarrow -A_{s-1} \qquad reject(-A_{s-1}) \leftarrow A_{s-1}$$

Other conditions and applications can be coded in this way. In particular, suitable rules can be used to enact preferences, to ensure compliance with integrity constraints or to ensure non-inertiality of fluents. Also, more complex contradiction removal criteria can be similarly coded. In all such cases, the semantic characterization of program updates would have to be adjusted accordingly to account for the change in their definition. However, pursuance of this topic is outside of the scope of the present paper.

7 Conclusions and Future Work

We defined a program transformation that takes two generalized logic programs P and U, and produces the updated logic program $P \oplus U$ resulting from the update of program P by U. We provided a complete characterization of the semantics of program updates $P \oplus U$ and we established their basic properties. Our approach generalizes the so called *revision programs* introduced in [9]. Namely, in the special case when the initial program is just a set of facts, our program update coincides with the justified revision of [9]. In the general case, when the initial program also contains rules, our program updates characterize precisely which of these rules remain valid by inertia, and which are rejected. We also showed how strong or *"classical"* negation can be easily incorporated into the framework of program updates.

With the introduction of dynamic program updates, we have extended program updates to ordered sets of logic programs (or modules). When this order is interpreted as a time order, dynamic program updates describe the evolution of a logic program which undergoes a sequence of modifications. This opens up the possibility of incremental design and evolution of logic programs, leading to the paradigm of *dynamic logic programming.* We believe that dynamic programming significantly facilitates *modularization* of logic programming, and, thus, modularization of nonmonotonic reasoning as a whole.

A specific application of dynamic logic programming that we intend to explore, is the evolution and maintenance of software specifications. By using logic programming as a specification language, dynamic programming provides the means of representing the evolution of software specifications.

However, ordered sets of program modules need not necessarily be seen as just a temporal evolution of a logic program. Different modules can also represent different sets of priorities, or viewpoints of different agents. In the case of priorities, a dynamic program update specifies the exact meaning of the "union" of the modules, subject to the given priorities. We intend to further study the relationship between dynamic logic programming and other preference-based approaches to knowledge representation.

Although not explored in here, a dynamic program update can be queried not only about the current state but also about other states. If modules are seen as viewpoints of different agents, the truth of some A_s in $\bigoplus \mathcal{P}$ can be read as: A is true according to agent s in a situation where the knowledge of the $\bigoplus \mathcal{P}$ is "visible" to agent s.

We are in the process of generalizing our approach and results to the *3-valued* case, which will enable us to update programs under the well-founded semantics. We have already developed a working implementation for the 3-valued case with top-down querying.

Our approach to program updates has grown out of our research on representing non-monotonic knowledge by means of logic programs. We envisage enriching it in the near future with other dynamic programming features, such as abduction and contradiction removal. Among other applications that we intend to study are productions systems modelling, reasoning about concurrent actions and active and temporal databases.

Acknowledgements

This work was partially supported by PRAXIS XXI project MENTAL, by JNICT project ACROPOLE, by the National Science Foundation grant # IRI931-3061, and a NATO scholarship while the L. M. Pereira was on sabbatical leave at the Department of Computer Science, University of California, Riverside. The work of J. A. Leite was supported by PRAXIS Scholarship no. BD/13514/97.

References

[1] J. J. Alferes, L. M. Pereira. *Update-programs can update programs.* In J. Dix, L. M. Pereira and T. Przymusinski, editors, Selected papers from the ICLP'96 Workshop NMELP'96, vol. 1216 of LNAI, pages 110-131. Springer-Verlag, 1997.

[2] J. J. Alferes, L. M. Pereira and T. Przymusinski. *Strong and Explicit Negation in Non-Monotonic Reasoning and Logic Programming.* In J. J. Alferes, L. M. Pereira and E. Orlowska, editors, JELIA '96, volume 1126 of LNAI, pages 143-163. Springer-Verlag, 1996.

[3] M. Gelfond and V. Lifschitz. *The stable model semantics for logic programming.* In R. Kowalski and K. A. Bowen. editors. 5th International Logic Programming Conference, pages 1070-1080. MIT Press, 1988.

[4] M. Gelfond and V. Lifschitz. *Logic Programs with classical negation.* In Warren and Szeredi, editors, 7th International Logic Programming Conference, pages 579-597. MIT Press, 1990.

[5] H. Katsuno and A. Mendelzon. *On the difference between updating a knowledge base and revising it.* In James Allen, Richard Fikes and Erik Sandewall, editors, Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference (KR91), pages 230-237, Morgan Kaufmann 1991.

[6] João A. Leite. *Logic Program Updates.* M.Sc. Dissertation, Universidade Nova de Lisboa, 1997.

[7] J. A. Leite and L. M. Pereira. *Generalizing updates: from models to programs.* In LPKR'97: ILPS'97 Workshop on Logic Programming and Knowledge Representation, Port Jefferson, NY, USA, October 13-16, 1997.

[8] V. Lifschitz and T. Woo. *Answer sets in general non-monotonic reasoning (preliminary report).* In B. Nebel, C. Rich and W. Swartout, editors, Principles of Knowledge Representation and Reasoning, Proceedings of the Third International Conference (KR92), pages 603-614. Morgan-Kaufmann, 1992

[9] V. Marek and M. Truszczynski. *Revision specifications by means of programs.* In C. MacNish, D. Pearce and L. M. Pereira, editors, JELIA '94, volume 838 of LNAI, pages 122-136. Springer-Verlag, 1994.

[10] T. Przymusinski and H. Turner. *Update by means of inference rules.* In V. Marek, A. Nerode, and M. Truszczynski, editors, LPNMR'95, volume 928 of LNAI, pages 156-174. Springer-Verlag, 1995.

[11] M. Winslett. *Reasoning about action using a possible models approach.* In Proceeding of AAAI'88, pages 89-93. 1988.

Qualitative Spatio/ Temporal Reasoning

Foundations of Spatioterminological Reasoning with Description Logics

Volker Haarslev, Carsten Lutz, Ralf Möller
Computer Science Department
University of Hamburg
Vogt-Kölln-Str. 30
22527 Hamburg, Germany

Abstract

This paper presents a method for reasoning about spatial objects and their qualitative spatial relationships. In contrast to existing work, which mainly focusses on reasoning about qualitative spatial relations alone, we integrate quantitative and qualitative information with terminological reasoning. For spatioterminological reasoning we present the description logic $\mathcal{ALCRP}(\mathcal{D})$ and define an appropriate concrete domain $\mathcal{D}$ for polygons. The theory is motivated as a basis for knowledge representation and query processing in the domain of deductive geographic information systems.

1 Introduction

Qualitative relations play an important role in formal reasoning systems that can be part of, for instance, geographic information systems (GIS). In this context, inferences about spatial relations should not be considered in isolation but should be integrated with formal inferences about structural descriptions of domain objects (e.g. automatic consistency checking and classification) and inferences about quantitative data. In our opinion, the abstractions provided by qualitative spatial relations can be interpreted as an interface from a conceptual model about the world to quantitative spatial data representing spatial information about domain objects. The combination of formal conceptual and spatial reasoning serves as a theoretical basis for knowledge representation in GIS and can be used to solve important application problems. Continuing our work presented in [11] and [13] we demonstrate the importance of terminological inferences with spatial relations in the domain of map databases and spatial query processing. In order to answer a query, concept terms are computed on the fly and must be checked for consistency. Furthermore, we assume that computed concept terms must be automatically inserted into the subsumption hierarchy of a knowledge base.

For formalizing reasoning about spatial structures many theories have been published (see e.g. [27] for an overview). Ignoring decidability, Borgo et al. [5] have developed a first order theory of space which formalizes different aspects such as mereology etc. An algebraic theory about space has been proposed in [21]. The well-known RCC theory [6] also formalizes *qualitative* reasoning about space. While first axiomatizations used first-order logic, recently, the spatial relations used in RCC have been defined in terms of intuitionistic logic and propositional modal logic [4]. Although qualitative reasoning with RCC can be used in many applications, in GIS also conceptual knowledge combined with quantitative data has to be considered. Therefore, another approach is required.

In order to adequately support decidable reasoning (i) about qualitative relations between spatial regions and (ii) about properties of quantitative data, we extend the description logic $\mathcal{ALC}(\mathcal{D})$ [2]. The main idea of our approach is to deal with spatial objects and their relations using predicates over concrete domain objects (see below for a formal introduction) and to deal with knowledge about abstract domain objects using the well-known description logic theory. Although description logics in general, and $\mathcal{ALC}$ in particular, are known to be strongly related to propositional modal logics [26, 7], it is not clear how Bennett's modal logic can be extended to handle conceptual modeling and quantitative data.

Extending the work on $\mathcal{ALC}(\mathcal{D})$, we have developed a new description logic called $\mathcal{ALCRP}(\mathcal{D})$ [16] in order to provide a foundation for spatioterminological reasoning with description logics ($\mathcal{RP}$ stands for role definitions based on predicates). The part of our theory dealing with spatial relations is based on a set of topological relations in analogy to Egenhofer [8] or RCC-8 [22]. The goal was to develop a description logic that provides modeling constructs which can be used to represent topological relations as defined roles. In a specific domain model, roles representing topological relations can be defined based on properties between concrete objects which, in turn, are associated to individuals via specific features. Thus, $\mathcal{ALCRP}(\mathcal{D})$ provides role terms that refer to predicates over a concrete domain. With these constructs $\mathcal{ALCRP}(\mathcal{D})$ extends the expressive power of $\mathcal{ALC}(\mathcal{D})$ (for a comparison, see [16]). However, in order to ensure termination of the satisfiability algorithm, we impose restrictions on the syntactic form of the set of terminological axioms. Although modeling is harder, the restrictions on terminologies ensure decidability of the language.

In contrast to our earlier work presented in [13], [10] and [14] where topological relations are used as primitives in the sense of logic, we extend the treatment of topological relations with respect to terminological reasoning. Thus, the theory presented in this paper allows one to detect both inconsistencies and implicit information in formal conceptual models for spatial domain objects (for a longer introduction, see [16], [18], [12], and [11]).

The paper is structured as follows. In the second section we introduce the language $\mathcal{ALCRP}(\mathcal{D})$. Unfortunately, without restrictions the satisfiability problem for $\mathcal{ALCRP}(\mathcal{D})$ is undecidable. We give a proof in the second section. Afterwards, we discuss the notion of a restricted terminology. A proof for the decidability of the satisfiability problem for restricted $\mathcal{ALCRP}(\mathcal{D})$ terminologies is sketched. Section 3 discusses an extended example for applying $\mathcal{ALCRP}(\mathcal{D})$ to spatioterminological reasoning. The conclusion in Section 4 demonstrates the general significance of $\mathcal{ALCRP}(\mathcal{D})$ by pointing out its applicability to other important reasoning problems such as terminological reasoning about temporal relations.

2 The Description Logic $\mathcal{ALCRP}(\mathcal{D})$

The description logic $\mathcal{ALC}(\mathcal{D})$ as defined in [2] supports the representation of concrete knowledge by incorporating concrete domains. The concrete domain approach can be used to integrate conceptual knowledge with concrete spatial knowledge. However, to define meaningful spatial concepts, it is also necessary to represent qualitative spatial relations and to exploit their various properties for reasoning. Since quantification over these spatial relations is also needed for modeling with a description logic formalism, they should be represented as roles. Furthermore, the formalism has to ensure that knowledge about qualitative relations is represented in a way that is consistent with quantitative spatial knowledge. In this section, the description logic $\mathcal{ALCRP}(\mathcal{D})$ is introduced. $\mathcal{ALCRP}(\mathcal{D})$ extends $\mathcal{ALC}(\mathcal{D})$ by a role-forming operator which is based on concrete domain predicates. The new operator allows the definition of roles with very complex properties and provides a close coupling of roles with concrete domains.

2.1 The Formalism

First, the concept language is introduced. Concrete domains are defined by Baader and Hanschke [2] as follows.

Definition 1. A *concrete domain* $\mathcal{D}$ is a pair $(\Delta_\mathcal{D}, \Phi_\mathcal{D})$, where $\Delta_\mathcal{D}$ is a set called the domain, and $\Phi_\mathcal{D}$ is a set of predicate names. Each predicate name P from $\Phi_\mathcal{D}$ is associated with an arity n, and an n-ary predicate $P^\mathcal{D} \subseteq \Delta^n_\mathcal{D}$. A concrete domain $\mathcal{D}$ is called *admissible* iff (1) the set of its predicate names is closed under negation and contains a name for $\Delta_\mathcal{D}$, (2) the satisfiability problem for finite conjunctions of predicates is decidable.

In the following we define role and concept terms in $\mathcal{ALCRP}(\mathcal{D})$.

Definition 2. Let R and F be disjoint sets of role and feature names[1], respectively. Any element of $\mathsf{R} \cup \mathsf{F}$ is an *atomic* role term. A composition of features (written $f_1 f_2 \cdots$) is called a feature chain. A simple feature can be viewed as a feature chain of length 1. If $P \in \Phi_\mathcal{D}$ is a predicate name with arity $n + m$ and $u_1, \ldots, u_n$ as well as $v_1, \ldots, v_m$ are feature chains, then the expression $\exists(u_1, \ldots, u_n)(v_1, \ldots, v_m).P$ (*role-forming predicate restriction*) is a *complex* role term. Let S be a role name and let T be a role term. Then $S \doteq T$ is a *terminological axiom*.

Definition 3. Let C be a set of concept names which is disjoint to R and F. Any element of C is a *concept term* (*atomic* concept term). If C and D are concept terms, R is a role term, $P \in \Phi_\mathcal{D}$ is a predicate name with arity n, and $u_1, \ldots, u_n$ are feature chains, then the following expressions are also concept terms: $C \sqcap D$ (*conjunction*), $C \sqcup D$ (*disjunction*), $\neg C$ (*negation*), $\exists R.C$ (*exists restriction*), $\forall R.C$ (*value restriction*), and $\exists u_1, \ldots, u_n.P$ (*predicate exists restriction*). For all kinds of exists and value restrictions, the role term or the list of feature chains may be written in parentheses. Let A be a concept name and let D be a concept term. Then $A \doteq D$ is a terminological axiom as well. A finite set of terminological axioms $\mathcal{T}$

[1]In the following, the notion *feature* is used as a synonym for feature name.

is a *terminology* or *TBox* if no concept or role name in $\mathcal{T}$ appears more than once on the left hand side of a definition and, furthermore, if no cyclic definitions are present.

We can now assign a meaning to $\mathcal{ALCRP}(\mathcal{D})$ concept terms by giving a set-theoretic semantics as usual.

Definition 4. An *interpretation* $\mathcal{I} = (\Delta_\mathcal{I}, \cdot^\mathcal{I})$ consists of a set $\Delta_\mathcal{I}$ (the abstract domain) and an interpretation function $\cdot^\mathcal{I}$. The sets $\Delta_\mathcal{D}$ and $\Delta_\mathcal{I}$ must be disjoint. The interpretation function maps each concept name C to a subset $C^\mathcal{I}$ of $\Delta_\mathcal{I}$, each role name R to a subset $R^\mathcal{I}$ of $\Delta_\mathcal{I} \times \Delta_\mathcal{I}$, and each feature name f to a partial function $f^\mathcal{I}$ from $\Delta_\mathcal{I}$ to $\Delta_\mathcal{D} \cup \Delta_\mathcal{I}$, where $f^\mathcal{I}(a) = x$ will be written as $(a, x) \in f^\mathcal{I}$. If $u = f_1 \cdots f_n$ is a feature chain, then $u^\mathcal{I}$ denotes the composition $f_1^\mathcal{I} \circ \cdots \circ f_n^\mathcal{I}$ of the partial functions $f_1^\mathcal{I}, \ldots, f_n^\mathcal{I}$. Let the symbols C, D, R, P, $u_1, \ldots, u_m$, and $v_1, \ldots, v_m$ be defined as in Definition 2 and 3, respectively. Then the interpretation function can be extended to arbitrary concept and role terms as follows:

$$\begin{aligned}
(C \sqcap D)^\mathcal{I} &:= C^\mathcal{I} \cap D^\mathcal{I} \\
(C \sqcup D)^\mathcal{I} &:= C^\mathcal{I} \cup D^\mathcal{I} \\
(\neg C)^\mathcal{I} &:= \Delta_\mathcal{I} \setminus C^\mathcal{I} \\
(\exists R.C)^\mathcal{I} &:= \{a \in \Delta_\mathcal{I} \mid \exists b \in \Delta_\mathcal{I} : \\
&\qquad (a, b) \in R^\mathcal{I}, b \in C^\mathcal{I}\} \\
(\forall R.C)^\mathcal{I} &:= \{a \in \Delta_\mathcal{I} \mid \forall b \in \Delta_\mathcal{I} : \\
&\qquad (a, b) \in R^\mathcal{I} \rightarrow b \in C^\mathcal{I}\} \\
(\exists u_1, \ldots, u_n.P)^\mathcal{I} &:= \{a \in \Delta_\mathcal{I} \mid \exists x_1, \ldots, x_n \in \Delta_\mathcal{D} : \\
&\qquad (a, x_1) \in u_1^\mathcal{I}, \ldots, (a, x_n) \in u_n^\mathcal{I}, \\
&\qquad (x_1, \ldots, x_n) \in P^\mathcal{D}\} \\
(\exists (u_1, \ldots, u_n)(v_1, \ldots, v_m).P)^\mathcal{I} &:= \\
&\{(a, b) \in \Delta_\mathcal{I} \times \Delta_\mathcal{I} \mid \\
&\quad \exists x_1, \ldots, x_n, y_1, \ldots, y_m \in \Delta_\mathcal{D} : \\
&\quad (a, x_1) \in u_1^\mathcal{I}, \ldots, (a, x_n) \in u_n^\mathcal{I}, \\
&\quad (b, y_1) \in v_1^\mathcal{I}, \ldots, (b, y_m) \in v_m^\mathcal{I}, \\
&\quad (x_1, \ldots, x_n, y_1, \ldots, y_m) \in P^\mathcal{D}\}
\end{aligned}$$

An interpretation $\mathcal{I}$ is a *model* of a TBox $\mathcal{T}$ iff it satisfies $A^\mathcal{I} = D^\mathcal{I}$ for all terminological axioms $A \doteq D$ in $\mathcal{T}$. A concept term C *subsumes* a concept term D w.r.t. a TBox $\mathcal{T}$ (written $D \preceq_\mathcal{T} C$), iff $D^\mathcal{I} \subseteq C^\mathcal{I}$ for all models $\mathcal{I}$ of $\mathcal{T}$. A concept term C is *satisfiable* w.r.t. a TBox $\mathcal{T}$ iff there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $D^\mathcal{I} \neq \emptyset$.

The basic reasoning service for a description logic formalism is computing the subsumption relationship. This inference is needed in the TBox to build a hierarchy of concepts w.r.t. specificity. Subsumption and satisfiability can be mutually reduced to each other since $C \preceq_\mathcal{T} D$ iff $C \sqcap \neg D$ is not satisfiable. The following definition introduces the assertional language of $\mathcal{ALCRP}(\mathcal{D})$, which can be used to represent knowledge about individual worlds.

Definition 5. Let O_D and O_A be two disjoint sets of object names. If C is a concept term, R a role term, f a feature name, P a predicate name with arity n, a and b are elements of O_A and x, and $x_1, \ldots, x_n$ are elements of O_D, then the following expressions are *assertional axioms.*

$$a : C,\ (a, b) : R,\ (a, x) : f,\ (x_1, \ldots, x_n) : P$$

A finite set of assertional axioms is called *ABox*. An *interpretation* for the concept language can be extended to the assertional language by additionally mapping every object name from O_A to a single element of $\Delta_\mathcal{I}$ and every object name from O_D to a single element from $\Delta_\mathcal{D}$. We assume that the unique name assumption does not hold, that is $a^\mathcal{I} = b^\mathcal{I}$ may hold even if $a \neq b$. An interpretation satisfies an assertional axiom

$$\begin{gathered}
a : C \text{ iff } a^\mathcal{I} \in C^\mathcal{I}, \quad (a, b) : R \text{ iff } (a^\mathcal{I}, b^\mathcal{I}) \in R^\mathcal{I}, \\
(a, x) : f \text{ iff } f^\mathcal{I}(a^\mathcal{I}) = x^\mathcal{I}, \\
(x_1, \ldots, x_n) : P \text{ iff } (x_1^\mathcal{I}, \ldots, x_n^\mathcal{I}) \in P^\mathcal{D}
\end{gathered}$$

An interpretation is a *model* of an ABox $\mathcal{A}$ w.r.t. a TBox $\mathcal{T}$, iff it is a model of $\mathcal{T}$ and furthermore satisfies all assertional axioms in $\mathcal{A}$. An ABox is *consistent* w.r.t. a TBox $\mathcal{T}$ iff it has a model.

The ABox consistency problem is to decide whether a given ABox $\mathcal{A}$ is consistent w.r.t. a TBox $\mathcal{T}$. Satisfiability of concept terms can be reduced to ABox consistency as follows: A concept term C is satisfiable iff the ABox $\{a : C\}$ is consistent. In the next section we show that the reasoning problems just introduced are undecidable if the full logic $\mathcal{ALCRP}(\mathcal{D})$ is considered.

2.2 Undecidability of the Full Logic

In the following we prove the undecidability of reasoning in $\mathcal{ALCRP}(\mathcal{D})$ if no restrictions are posed on terminologies. The undecidability of the satisfiability problem is shown and this implies subsumption and ABox consistency are also undecidable.

Theorem 6. *The problem whether an $\mathcal{ALCRP}(\mathcal{D})$ concept term C is satisfiable w.r.t. a TBox $\mathcal{T}$ is undecidable.*

Proof. The proof works by reducing the Post Correspondence Problem (PCP) to the satisfiability problem for $\mathcal{ALCRP}(\mathcal{D})$ concept terms and is similar to the reduction given in [3]. The PCP is defined as follows. Given a nonempty finite set $S = \{(l_i, r_i); i = 1, \ldots, m\}$, where the l_i and r_i are words over an alphabet Σ, a *solution of* S is a sequence of indices $i_1, \ldots, i_k$ with $k \geq 1$ such that the left concatenation $w_l = l_{i_1} \cdots l_{i_k}$ and the right concatenation $w_r = r_{i_1} \cdots r_{i_k}$ denote the same word. The PCP is known to be undecidable if Σ contains at least two symbols [20].

For the reduction, the words over Σ are encoded as natural numbers which are then represented as concrete domain objects. The elements of Σ are interpreted as digits of numbers at base B, where $B := |\Sigma| + 1$. $\overline{w}$ denotes the nonnegative integer at base 10 which the (nonempty) word w represents at base B. $w \mapsto \overline{w}$ is a $1-1$-mapping from Σ^* into the set of nonnegative integers. Concatenation is encoded as an operation over natural numbers that can be captured by a concrete domain predicate: If vw is the concatenation of two words $v, w \in \Sigma^*$, then $\overline{vw} = \overline{v} * B^{|w|} + \overline{w}$, where $|w|$ is the length of the word w.

Now for every instance S of the PCP, an $\mathcal{ALCRP}(\mathcal{D})$ concept *C(S)* can be defined whose models have an infinite tree-like structure that encodes the set of possible solutions of the PCP instance S. Additionally, the concept is defined in a way such that it is satisfiable if and only if none of the possible solutions in fact is a solution. Hence, if the satisfiability of the concept *C(S)* could be decided then one would be able to decide the satisfiability of the PCP S. Since this is impossible, the satisfiability problem for $\mathcal{ALCRP}(\mathcal{D})$ concept terms must be undecidable. Let l, r, w_l, w_r and $f_1, \ldots, f_m$ be attributes and R be a role name, then for a given instance S of the PCP we define a concept *C(S)*:

$$
\begin{aligned}
C(S) \doteq\; & \exists w_l.\mathit{zero\text{-}p} \sqcap \exists w_r.\mathit{zero\text{-}p} \sqcap & (1)\\
& \sqcap_{i=1}^{m} \exists w_l, f_i \circ w_l.\mathit{cnstr\text{-}p}_l^i \sqcap & (2)\\
& \sqcap_{i=1}^{m} \exists w_r, f_i \circ w_r.\mathit{cnstr\text{-}p}_r^i \sqcap & (3)\\
& \forall R. \sqcap_{i=1}^{m} \exists w_l, f_i \circ w_l.\mathit{cnstr\text{-}p}_l^i \sqcap & (4)\\
& \forall R. \sqcap_{i=1}^{m} \exists w_r, f_i \circ w_r.\mathit{cnstr\text{-}p}_r^i \sqcap & (5)\\
& \forall R. \exists w_l, w_r.\mathit{notequal\text{-}p} & (6)
\end{aligned}
$$

$$R \doteq \exists (w_l, w_r)(w_l, w_r).\mathit{trans\text{-}p}$$

The predicates used in the definition of *C(S)* are defined as follows (l_i and r_i are the words used in the definition of the PCP problem, see above):

$$
\begin{aligned}
\mathit{trans\text{-}p}(a,b,c,d) &:= a < c \wedge b < d\\
\mathit{zero\text{-}p}(a) &:= a = 0\\
\mathit{cnstr\text{-}p}_l^i(a,b) &:= b = \overline{l_i} + a * B^{|l_i|}\\
\mathit{cnstr\text{-}p}_r^i(a,b) &:= b = \overline{r_i} + a * B^{|r_i|}\\
\mathit{notequal\text{-}p}(a,b) &:= a \neq b
\end{aligned}
$$

To complete the proof of theorem 6, we need to prove the following proposition.

Proposition 7. *The concept C(S) is satisfiable if and only if the PCP S has no solution.*

Proof. It has to be shown that (i) if *C(S)* is satisfiable then S has no solution and (ii) if S has no solution then *C(S)* is satisfiable.

The first point can be seen by examining the definition of *C(S)*. If *C(S)* is satisfiable then there exists an interpretation $\mathcal{I}$ with $C(S)^{\mathcal{I}} \neq \emptyset$. As intended, this interpretation encodes all possible sequences that could be a solution of the PCP S. It has the form of an infinite tree. Each node (abstract object) in the tree has m successors. The edges (attributes) to these successors are labeled with f_1 to f_m, respectively. The edges represent single indices and the nodes represent sequences of indices defined by the labels of the path from the root node. The role R is used as a "universal role" to propagate concepts to every node in the tree: every node in the tree is an R-role filler of the root-node. In lines 4 and 5, predicate-operators which use the predicate *cnstr-p* are propagated to every node in the tree. These concepts enforce the successors of each node. The concepts in lines 2 and 3 do the same for the root node. Each node in the tree has concrete fillers of the *wl* and *wr* attributes. These fillers are enforced by the concept terms in lines 1-5. The fillers of these attributes encode the two concatenations that are defined by the index sequence which the node represents. Line 6 enforces that for all objects in the tree the fillers of *wl* and *wr* (and thus the concatenations for all possible sequences) differ. From this it is clear that the concept *C(S)* can only be satisfiable if the PCP S has no solution.

To prove the second point we give an interpretation with $C(S)^{\mathcal{I}} \neq \emptyset$ for a given PCP S for which it is known that no solution exists. This interpretation has the same structure as described above.

$$\Delta_{\mathcal{I}} = \{a_{ij}; i \geq 0, 0 \leq j < m^i\};$$
$$\forall i \geq 0, 0 \leq j < m^i:$$
$$f_1^{\mathcal{I}}(a_{ij}) = a_{i+1\ j*m}, \ \ldots, \ f_m^{\mathcal{I}}(a_{ij}) = a_{i+1\ j*m+m-1},$$
$$w_l^{\mathcal{I}}(a_{ij}) = \overline{\phi_l(i,j)}, \quad w_r^{\mathcal{I}}(a_{ij}) = \overline{\phi_r(i,j)}$$

where ϕ_l and ϕ_r are two recursively defined concatenation functions (*concat* concatenates words and $\lfloor\rfloor$ denotes the *floor* function):

$$\phi_l(0,0) = \epsilon$$
$$\phi_r(0,0) = \epsilon$$
$$\phi_l(i,j) = concat(\phi_l(i-1, \lfloor j/m \rfloor), l_{j+1-(m*\lfloor j/m \rfloor)})$$
$$\phi_r(i,j) = concat(\phi_r(i-1, \lfloor j/m \rfloor), r_{j+1-(m*\lfloor j/m \rfloor)}).$$

□

As already noted, in [3] a similar proof is used to prove the undecidability of subsumption for $\mathcal{ALC}(\mathcal{D})$ extended by a transitivity operator for roles. The proof given there differs in that it uses a concept which is satisfiable if and only if the corresponding PCP has a solution. A transitive attribute is used to distribute concepts to any node. This cannot be done in $\mathcal{ALCRP}(\mathcal{D})$. However, with a role-forming predicate operator we can define a transitive role and, therefore, we have to use a different proof strategy (see [17] for further details).

The new role-forming operator plays an important role in the definition of the concept *C(S)*. As already noted, the complex role R is used as a universal role, i.e. it connects the root node of the tree to all other nodes. Using value restriction over R, a concept term which contains exists restrictions creating new succeeding nodes is propagated to every node of the tree. It is this mechanism that enforces the infinite tree structure of the model of *C(S)*. In the following section a set of restrictions for terminologies is developed. The idea is that these restrictions prohibit the definition of "dangerous" concepts such as *C(S)*.

2.3 Restricted Terminologies

The analysis of the concept *C(S)* at the end of the last section gives a first idea on how to avoid undecidability. Complex roles seem to cause problems if they are used in certain combinations with exists and value restrictions. A more thorough analysis reveals that there exist at least two options for restricting the language such that reasoning becomes decidable. First, restrictions could be posed on the structure of the concrete domain predicates. This is not very promising since all interesting applications, e.g. modeling spatial or temporal relations, require fairly complex predicate definitions. There is strong evidence that reasoning in $\mathcal{ALCRP}(\mathcal{D})$ will be undecidable if appropriate predicates are defined. Second, some critical combinations of concept-forming operators could be restricted. We will pursue the second approach. Before we can introduce our structurally restricted terminologies, we have to make some technical definitions.

A concept term C is said to be *unfolded* w.r.t. a TBox $\mathcal{T}$ iff none of the concept and role names used in the concept term occur on the left side of any terminological axiom in $\mathcal{T}$. Any concept term can be transformed into an unfolded form by iteratively replacing concept and role names by their defining terms. This algorithm terminates since the terminology is required to be acyclic. Any unfolded concept term can then be transformed to an equivalent one in *negation normal form* (NNF). An unfolded concept term is said to be in NNF iff negation occurs only in front of concept names. The transformation to NNF can be done by iteratively applying transformation rules that propagate the negations "down" to the atomic concepts. For example, $\neg(C \sqcap D)$ has to be transformed to $\neg C \sqcup \neg D$. We omit details since the transformation rules are the same as for $\mathcal{ALC}(\mathcal{D})$ [2]. We are now ready to define restricted terminologies.

Definition 8. A concept term X is called *restricted* w.r.t. a TBox $\mathcal{T}$ iff its equivalent X' which is unfolded w.r.t. $\mathcal{T}$ and in NNF fulfills the following conditions:[2]

(1) For any subconcept term C of X' that is of the form $\forall R_1.D$ where R_1 is a complex role term, D does not contain any terms of the form $\exists R_2.E$ where R_2 is also a complex role term.

(2) For any subconcept term C of X' that is of the form $\exists R_1.D$ where R_1 is a complex role term, D does not contain any terms of the form $\forall R_2.E$ where R_2 is also a complex role term.

(3) For any subconcept term C of X' that is of the form $\forall R.D$ or $\exists R.D$ where R is a complex role term, D contains only predicate exists restrictions that (i) quantify over attribute chains of length 1 and (ii) are not contained inside any value and exists restrictions that are also contained in D.

A terminology is called restricted iff all concept terms appearing on the right-hand side of terminological axioms in $\mathcal{T}$ are restricted w.r.t. $\mathcal{T}$. An ABox $\mathcal{A}$ is called restricted w.r.t. a TBox $\mathcal{T}$ iff $\mathcal{T}$ is restricted and all concept terms used in $\mathcal{A}$ are restricted w.r.t. the terminology $\mathcal{T}$.

Consider for example the following three very simple terminologies that are already in unfolded NNF. None of them is restricted because they all violate one of the above conditions. Let C and D be concept names, R_a be an atomic role term, R_c be a complex role term, f be a feature, and u be a feature chain with a length greater than 1.

[2]For technical reasons, we assume that a concept term is a subconcept term of itself.

$$\mathcal{T}_1 : \{C \doteq \forall R_c.\exists R_c.D\},$$
$$\mathcal{T}_2 : \{C \doteq \exists R_c.\exists u.P\},$$
$$\mathcal{T}_3 : \{C \doteq \forall R_c.\forall R_a.\exists f.P\}$$

We can now examine the decidability of the standard reasoning problems with respect to restricted terminologies of $\mathcal{ALCRP}(\mathcal{D})$.

Theorem 9. *The ABox consistency problem for restricted $\mathcal{ALCRP}(\mathcal{D})$ ABoxes is decidable.*

We prove this by giving an algorithm which is sound and complete.

2.4 The Tableau Calculus

The algorithm is a standard tableau-based one as it is used for first order or modal logics as well as other description logics. To decide the satisfiability of a concept C, the algorithm starts with an initial ABox $\mathcal{A}_0 := \{a : C\}$ and then iteratively applies completion rules creating one or more descendant ABoxes. Thus, rule application constructs a tree of ABoxes Υ. Finally, either all ABoxes that are leaves of Υ happen to be contradictory which means that C is not satisfiable or a non-contradictory ABox is obtained to which no more completion rules are applicable. In the latter case, the ABox mentioned is called *complete* and defines a model for C. The purpose of the completion rules can be understood as making implicit facts explicit. The algorithm can be considered as a step-by-step model construction.

The rule set is an extension of the one used for deciding ABox consistency in $\mathcal{ALC}(\mathcal{D})$ (see [2]). Before the rules can be given, some technical terms need to be defined. Let $\mathcal{A}$ be an ABox, R be a role term, a and b be object names from O_A, γ be a symbol that is not element of O_D, u be a feature chain $f_1 \circ \cdots \circ f_k$, and let $u_1, \ldots, u_n$ and $v_1, \ldots, v_m$ be arbitrary feature chains. For convenience we define three functions as follows:

$\mathit{filler}_{\mathcal{A}}(a, u) :=$

x where $x \in O_D$ such that
$\exists b_1, \ldots, b_{k-1} \in O_A :$
$((a, b_1) : f_1 \in \mathcal{A}, \ldots, (b_{k-1}, x) : f_k \in \mathcal{A})$
γ if no such x exists.

$\mathit{chain}_{\mathcal{A}}(a, x, u) := \{(a, c_1) : f_1, \ldots, (c_{k-1}, x) : f_k\}$
where the $c_1, \ldots, c_{k-1} \in O_A$ are not used in $\mathcal{A}$.

$\mathit{filler?}_{\mathcal{A}}(a, b, R) :=$

true if $(a, b) : R \in \mathcal{A}$
true if R is of the form
$\exists(u_1, \ldots, u_n)(v_1, \ldots, v_m).P$ and
$\exists x_1, \ldots, x_n, y_1, \ldots, y_m \in O_D$ such that
$\mathit{filler}_{\mathcal{A}}(a, u_1) = x_1 \wedge \ \ldots \ \wedge \mathit{filler}_{\mathcal{A}}(a, u_n) = x_n \wedge$
$\mathit{filler}_{\mathcal{A}}(b, v_1) = y_1 \wedge \ \ldots \ \wedge \mathit{filler}_{\mathcal{A}}(b, v_m) = y_m \wedge$
$(x_1, \ldots, x_n, y_1, \ldots, y_m) : P \in \mathcal{A}$
false in all remaining cases.

An ABox $\mathcal{A}$ is said to contain a *fork* (for a feature f) if it contains the two axioms $(a, b) : f$ and $(a, c) : f$, where a and b are either both from O_A or O_D. A fork can be eliminated by replacing all occurrences of c in $\mathcal{A}$ with b. We assume that during rule application forks are eliminated as soon as they appear. Before any rules are applied to initial ABox $\mathcal{A}_0$, fork elimination also takes place. The completion rules can now be defined.

Definition 10. The following *completion rules* will replace an ABox $\mathcal{A}$ by a single ABox $\mathcal{A}'$ or by two ABoxes $\mathcal{A}'$ and $\mathcal{A}''$ (*descendants* of $\mathcal{A}$). In the following, C and D denote concept terms, R denotes a role term, and P denotes a predicate name from $\Phi_{\mathcal{D}}$. Let $f_1, \ldots, f_n$ denote feature names, and $u_1, \ldots, u_m$ as well as $v_1, \ldots, v_m$ denote feature chains. a and b denote object names from O_A.

R$\sqcap$ The conjunction rule.
Premise: $a : C \sqcap D \in \mathcal{A}, \quad a : C \notin \mathcal{A} \vee a : D \notin \mathcal{A}$
Consequence: $\mathcal{A}' = \mathcal{A} \cup \{a : C, \ a : D\}$

R$\sqcup$ The disjunction rule.
Premise: $a : C \sqcup D \in \mathcal{A}, \quad a : C \notin \mathcal{A} \wedge a : D \notin \mathcal{A}$
Consequence: $\mathcal{A}' = \mathcal{A} \cup \{a : C\}$, $\mathcal{A}'' = \mathcal{A} \cup \{a : D\}$

R$\exists$C The exists restriction rule.
Premise: $a : \exists R.C \in \mathcal{A}, \quad \neg\exists b \in O_A :$
$(\mathit{filler?}_{\mathcal{A}}(a, b, R) \ \wedge \ b : C \in \mathcal{A})$
Consequence: $\mathcal{A}' = \mathcal{A} \cup \{(a, b) : R \ , \ b : C\}$
where $b \in O_A$ is not used in $\mathcal{A}$.

R$\forall$C The value restriction rule.
Premise: $a : \forall R.C \in \mathcal{A}, \quad \exists b \in O_A :$
$(\mathit{filler?}_{\mathcal{A}}(a, b, R), \wedge \ b : C \notin \mathcal{A})$
Consequence: $\mathcal{A}' = \mathcal{A} \cup \{b : C\}$

R$\exists$P The predicate exists restriction rule.
Premise:
$a : \exists u_1, \ldots, u_n.P \in \mathcal{A}, \neg\exists x_1, \ldots, x_n \in O_D :$
$(\mathit{filler}_{\mathcal{A}}(a, u_1) = x_1 \wedge \ \ldots \ \wedge \mathit{filler}_{\mathcal{A}}(a, u_n) = x_n \wedge$
$(x_1, \ldots, x_n) : P \in \mathcal{A})$
Consequence:
$\mathcal{A}' = \mathcal{A} \cup \{(x_1, \ldots, x_n) : P\} \cup$

$chain_{\mathcal{A}}(a, x_1, u_1) \cup \ldots \cup chain_{\mathcal{A}}(a, x_n, u_n)$
where the objects $x_i \in O_D$ are not used in $\mathcal{A}$.

Rr∃P The complex role rule.
Premise:
$(a, b) : \exists(u_1, \ldots, u_n)(v_1, \ldots, v_m).P \in \mathcal{A}$,
$\neg\exists x_1, \ldots, x_n, y_1, \ldots, y_m \in O_D$:
$(filler_{\mathcal{A}}(a, u_1) = x_1 \wedge \ldots \wedge filler_{\mathcal{A}}(a, u_n) = x_n \wedge$
$filler_{\mathcal{A}}(b, v_1) = y_1 \wedge \ldots \wedge filler_{\mathcal{A}}(b, v_m) = y_m \wedge$
$(x_1, \ldots, x_n, y_1, \ldots, y_m) : P \in \mathcal{A})$
Consequence:
$\mathcal{A}' = \mathcal{A} \cup \{(x_1, \ldots, x_n, y_1, \ldots, y_m) : P\} \cup$
$chain_{\mathcal{A}}(a, x_1, u_1) \cup \ldots \cup chain_{\mathcal{A}}(a, x_n, u_n) \cup$
$chain_{\mathcal{A}}(b, y_1, v_1) \cup \ldots \cup chain_{\mathcal{A}}(b, y_m, v_m)$
where the objects $x_i \in O_D$ and $y_i \in O_D$ are not used in $\mathcal{A}$.

RChoose The choose rule.
Premise:
$a : \forall(\exists(u_1, \ldots, u_n)(v_1, \ldots, v_m).P).C \in \mathcal{A}$,
$\exists b \in O_A,\ x_1, \ldots, x_n, y_1, \ldots, y_m \in O_D$:
$(filler_{\mathcal{A}}(a, u_1) = x_1 \wedge \ldots \wedge filler_{\mathcal{A}}(a, u_n) = x_n \wedge$
$filler_{\mathcal{A}}(b, v_1) = y_1 \wedge \ldots \wedge filler_{\mathcal{A}}(b, v_m) = y_m \wedge$
$(x_1, \ldots, x_n, y_1, \ldots, y_m) : P \notin \mathcal{A} \wedge$
$(x_1, \ldots, x_n, y_1, \ldots, y_m) : \overline{P} \notin \mathcal{A})$
Consequence:
$\mathcal{A}' = \mathcal{A} \cup \{(x_1, \ldots, x_n, y_1, \ldots, y_m) : P\}$,
$\mathcal{A}'' = \mathcal{A} \cup \{(x_1, \ldots, x_n, y_1, \ldots, y_m) : \overline{P}\}$

The notion of a contradictory ABox still needs to be formally defined.

Definition 11. Let the same naming conventions be given as in Definition 10. Additionally, let f be a feature. An ABox $\mathcal{A}$ is called *contradictory* if any of the following *clash triggers* are applicable:
Primitive Clash: $a : C \in \mathcal{A}, \quad a : \neg C \in \mathcal{A}$
Feature Domain Clash: $(a, x) : f \in \mathcal{A}, \quad (a, b) : f \in \mathcal{A}$
All Domain Clash: $(a, x) : f \in \mathcal{A}, \quad a : \forall f.C \in \mathcal{A}$
Concrete Domain Clash:
$(x_1^{(1)}, \ldots, x_{n_1}^{(1)}) : P_1 \in \mathcal{A}, \ldots, (x_1^{(k)}, \ldots, x_{n_k}^{(k)}) : P_k \in \mathcal{A}$
and the corresponding conjunction $\bigwedge_{i=1}^{k} P_i(x^{(i)})$ is not satisfiable in $\mathcal{D}$. This can be decided because $\mathcal{D}$ is required to be admissible.

Similar completion rules can also be found in algorithms for related languages such as $\mathcal{ALC}(\mathcal{D})$. The clash triggers are identical to those of $\mathcal{ALC}(\mathcal{D})$. The new completion rules are Rr∃P and RChoose. The use of the *related* function is also new and is necessary because the new role-forming operator is introduced. In the following we will discuss only the novelties as compared to the rules needed for $\mathcal{ALC}(\mathcal{D})$. First, consider the rule R∀C. The use of the *related* function is necessary because not all role fillers might appear explicitly as constraints of the form $(a, b) : R$. If R is a complex role then b could be an R role filler of a, although there is no constraint of the above form. Two objects can be related simply because a predicate holds over the concrete fillers of feature chains starting from a and b, respectively. This can be seen by considering the semantics of the role-defining operator. The two possible types of role fillers (explicit and implicit) are both captured by the *related* function.

The meaning of the rule Rr∃P is straightforward. If it is known that two objects are related via a complex role, we can immediately create concrete objects as fillers of those feature chains being used in the definition of the complex role. It is then also known that the predicate used in the definition of the role holds over the newly created concrete objects and thus appropriate constraints can be added. This is what Rr∃P does.

The meaning of RChoose can be understood as follows. If a set of feature chains is used in the definition of a complex role R and (loosely spoken) there are the appropriate concrete fillers for two objects a and b for all the feature chains in this set, then b might be an R role filler of a. But unless there is an explicit constraint which states that the predicate P used in the definition of the role R holds (or does not hold), we do not know if this is really the case. So, if there is no such constraint, we have to try both alternatives and test whether P holds or does not hold. If any of these two alternatives is the wrong one, we will end up with a concrete domain clash in the corresponding branch of the ABox tree Υ. Like R⊔, an application of RChoose creates a branch in Υ.

A full proof of the soundness and completeness of the algorithm given above can be found in [16]. Soundness of the algorithm follows from the local soundness of the completion rules. Completeness can be proven by showing that any complete ABox computed by the algorithm defines a model for the initial ABox $\mathcal{A}_0$. However, proving termination is less straightforward. The restrictions posed on terminologies are required to ensure that all satisfiable concepts from the language are also satisfied by at least one finite model. If this finite model property does not hold, termination of the tableau algorithm can not be guaranteed.

We give a short sketch of the termination proof and then motivate the definition of the restrictions on concept terms. Each ABox that is computed by the application of completion rules can be mapped to a forest, i.e. a collection of trees, in the following way: Each element of O_A that is already present in $\mathcal{A}_0$ is the root of a tree. The edges and further nodes of the trees correspond to role filler relationships and abstract objects, respectively. But only the explicit role filler relationships introduced by the rules R∃C, R∃P and Rr∃P are taken into account. The number of the trees obviously doesn't grow during rule application. It can be shown that (i) infinite rule application implies infinite growth of at least one of the trees in at least one sequence of forests[3] and (ii) that there exist upper bounds for the degree and the depth of all the trees. From this it follows that the algorithm terminates. The restrictions on concept terms ensure the existance of the upper bound for the depth of the trees. For the detailed termination proof see [16].

As already noted at the end of section 2.2, in the case of unrestrictedness there are concept terms which are only satisfied by infinite models. To shed more light on details about the restrictions given in Definition 8, it is necessary to analyze how these concepts would be treated by the tableau calculus given in this section. Using a value restriction over a complex role $\forall R_c.C$, a concept term $C \doteq \exists R.D$ can be "propagated" to an object o_1. Then, the application of the R∃C rule creates an object o_2 with $(o_1, o_2) : R$. The concept term D can be crafted in a way such that concrete objects are generated as fillers of some attributes of o_2. This leads to the inference of a new role filler relationship between another object and the newly created object o_2. Again a value restriction is used to propagate an exists restriction along the role filler relationship just inferred to the object o_2. Thus, a cycle is obtained. To prevent this, one has to prevent the generation of concrete fillers for the attributes of o_2. Concrete objects are only created by the rules R∃P and Rr∃P. The rule R∃P can only be applied if there is a concept-forming predicate operator inside the concept term D. The rule Rr∃P can only be applied if the concept term C contains an exists restriction quantifying over a complex role. Summarizing, concept-forming predicate operators inside of value restrictions quantifying over complex roles and also nestings of value and exists restrictions which both quantify over complex roles have to be prohibited. Further elaboration yields the restrictions given in definition 8.

Corollary 12. *The subsumption problem and the satisfiability problem for $\mathcal{ALCRP}(\mathcal{D})$ concept terms are decidable w.r.t. terminologies for which the considered concept terms are restricted.*

Proof. This follows from Theorem 9 together with the reduction of subsumption to satisfiability and of satisfiability to ABox consistency (see Section 2.1). Please note that if the concept terms C and D are restricted w.r.t. a terminology $\mathcal{T}$, then the concept term $C \sqcap \neg D$ is also restricted w.r.t. $\mathcal{T}$ since the set of restricted concept terms is closed under negation. □

[3]A sequence of forrests corresponds to one path in Υ.

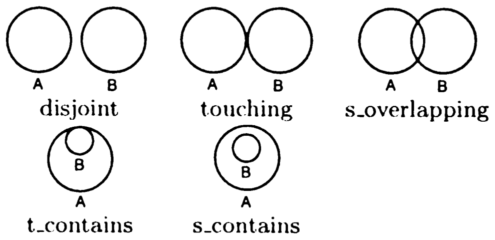


Figure 1: Elementary relations between two regions A and B. The inverses of t_contains and s_contains as well as the relation equal have been omitted.

In order use $\mathcal{ALCRP}(\mathcal{D})$ for knowledge representation in general and for spatial reasoning in particular, an admissible concrete domain must be defined. This will be discussed in the next section.

3 A Concrete Domain for Polygonal Space

Rather than dealing with arbitrary point sets in $\Re^2$, we restrict the predicates for the spatial domain to the description of polygons because efficient algorithms (e.g. the simplex procedure) are known for the polygon inclusion and polygon intersection problems. In accordance to Definition 1 we define the concrete domain $\mathcal{D}_\mathcal{P}$ as consisting of a set $\Delta_{\mathcal{D}_\mathcal{P}}$ of polygons and a set $\Phi_{\mathcal{D}_\mathcal{P}}$ of predicate names. Polygons are defined as usual as a list of polylines, i.e. polygons describe point sets that are not necessarily internally connected.

3.1 Predicates for Qualitative Spatial Relationships

The set $\Phi_{\mathcal{D}_\mathcal{P}}$ contains the names of eight elementary binary predicates (equal, disjoint, touching, strictly_overlapping, tangentially_contains/tangentially_inside, strictly_contains/strictly_inside) representing spatial relationships as illustrated in Figure 1. In analogy to Egenhofer [8] or RCC-8 [22] we have given a formal definition of the elementary relations in [11]. The definition is based on the interior, the complement and the boundary of spatial objects (point sets). The interior of a set λ is defined to be the union of all open sets in λ. The boundary of a polygon, i.e. the intersection of the closure of the interior and the closure of the complement of an object, is defined by the "border polyline" of the polygon.

For convenience, we extend the set $\Phi_{\mathcal{D}_\mathcal{P}}$ by names for so-called composite predicates consisting of disjunctions of elementary predicates. For instance, in this paper we use the universal relation spatially_related and the relation generally_inside (g_inside ≡ t_inside ∨ s_inside ∨ equal). We define one-place predicates which are denoted as $\mathsf{sr_p}$ where sr is an elementary or composite predicate and p is a concrete object representing a polygon constant.

3.2 Satisfiability of Conjunctions of Predicates

The admissibility criterion for $\mathcal{D}_\mathcal{P}$ concerns the satisfiability of finite conjunctions of (possibly negated) predicate terms. Negated unary and binary terms can be resolved into disjunctions of elementary spatial predicates by simple syntactic transformations because the elementary predicates are mutually exclusive and exhaustive. Therefore, we can restrict our analysis to conjunctions of unnegated binary terms $\bigvee_{j=1}^{k} \mathsf{ep}^j(x,y)$ where $1 \leq k \leq 8$ and $\mathsf{ep}^j \in \mathcal{EP}$.

Consistency of a conjunction of binary predicate terms is usually considered as a binary constraint satisfaction problem. In this view, a conjunction is represented as a constraint network whose nodes are defined by variable names and whose edges are labeled by relation sets representing disjunctions of relations between a pair of nodes. A standard technique for deciding the satisfiability of such a network is the *3-consistency* or *path consistency* method that is based on a composition table. This table defines the composition of spatial relations, for instance it has to hold that s_inside∘s_inside = s_inside (see above). In other words, a composition table directly encodes so-called *3-consistent* or *path consistent* spatial relations between three regions, e.g. $\mathsf{s\_inside}(A,B) \wedge \mathsf{s\_inside}(B,C) \Rightarrow \mathsf{s\_inside}(A,C)$.

However, in general, path consistency is not a sufficient criterion for consistency. Thus, an additional step is required to ensure *global* consistency. Algorithms for solving these constraint problems are discussed in [15] and [19, 24]. According to Nebel and Renz [19, 24], the worst case complexity depends on the relations (disjunctions of base relations) actually encountered in a constraint network. The problem of achieving global consistency is in $\mathcal{NP}$. However, Nebel and Renz [19, 24] showed that for certain subsets of $\mathcal{EP}$ global consistency is equivalent to path consistency. These findings can be used to speed up the verification of consistency (Ladkin and Reinefeld [15] also discuss speedup techniques).

Grigni et al. [9] consider objects that are internally connected regions in the plane and propose two notions of satisfiability, relational consistency and realizability. First, a conjunction may violate the so-called *relational consistency* criterion which is identical to the global consistency of a (spatial) constraint network (see above). The full form of satisfiability is called *realizability* and is related to *planarity*. A relationally consistent conjunction may violate realizability if planar regions are declared to be disjoint from one another (for examples see [9]). Thus, according to their semantics (regions are internally connected), there are conjunctions of predicates that are relationally consistent but not realizable in the plane.

In our approach, the geometric relationship between two concrete polygons (or its border polylines) directly corresponds to the qualitative spatial relationship of the objects. Hence, topological reasoning can be realized by inference processes based on composition tables when all spatial relations implicitly given by concrete polygons are computed (non-concrete spatial objects are not assumed to be internally connected). The work of Renz [23] shows that, in the case of regions that are not necessarily internally connected (open set semantics), satisfiability of RCC-8 constraint systems implies realizability. Thus, an algorithm for the concrete domain satisfiability test can be divided into three main steps:

1. Negated predicate terms are replaced by the corresponding disjunction of elementary predicate terms. Afterwards, every conjunct consists of either a single term $\mathsf{sr}(x,y)$ or a disjunction $\bigvee_{i=1}^{k} \mathsf{sr}_i(x,y)$ with $\mathsf{sr}, \mathsf{sr}_i \in \mathcal{ER}$, $1 \leq k \leq 8$, and all $\mathsf{sr}_i(x,y)$ are involved with the same pair of objects. For instance, $\neg\mathsf{g\_inside}(x,y)$ is replaced by $\mathsf{t\_contains}(x,y) \vee \mathsf{s\_contains}(x,y) \vee \mathsf{s\_overlapping}(x,y) \vee \mathsf{touching}(x,y) \vee \mathsf{disjoint}(x,y)$. Furthermore, for each conjunct a new conjunct representing the inverted relation term is added. For example, for the term $\mathsf{t\_contains}(x,y)$ we add $\mathsf{t\_inside}(y,x)$.
2. The one-place predicates sr_p^i introduce concrete polygons. With the help of standard algorithms from computational geometry we compute the topological base relation between each pair of concrete polygons. In accordance to our definition of the spatial relations, this problem basically can be reduced to the intersection test for two polygons. For each pair of polygons, the topological information is added to the constraint system with a corresponding predicate. From now on, concrete polygons are treated as variables.
3. In the third step, relational consistency is verified. Thus, a globally consistent solution must be computed (path-consistency may be realized as a pruning step). If no globally consistent solution can be found, return "not satisfiable", otherwise return "satisfiable."

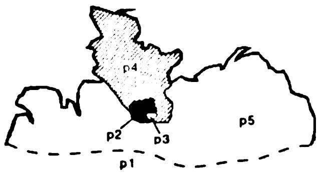


Figure 2: A sketch of the northern part of Germany with polygons for Germany (p_1), Northern Germany (p_5), the federal states Schleswig-Holstein (p_4) and Hamburg (p_2). Polygon p_2 is touching p_4 and p_3 is t_inside p_2.

In summary, we showed that the concrete domain $\mathcal{P}$ is admissible. The next section discusses the application of the algorithm using an application example.

3.3 Examples for Spatioterminological Reasoning

How can conrete domain predicates be used to support spatioterminological inferences? First of all, as an ontological commitment we assume that each domain object is associated with its spatial representation via the attribute has_area. We would like to define predicates that restrict the role fillers for has_area to be specific spatial regions. For instance, without being told explicitly, the inference system should automatically classify a region in Hamburg also as a Northern German region (see Figure 2).

For instance, the one-place predicate $\mathsf{g\_inside}_{\mathsf{p}}$ can be used as follows. Using the concept-forming predicate operator $\exists f . P$ (see above), we define restrictions for fillers of the feature (or attribute) has_area of a region in Northern Germany, for a district of the city of Hamburg, etc. The restrictedness criterion (cf. Definition 8) for the following two TBox axioms is trivially fulfilled because they contain no nested exists or all quantifiers.

$$\mathbf{northern\_german\_region} \doteq \exists\, \mathsf{has\_area} . \mathsf{g\_inside}_{\mathsf{p_5}}$$
$$\mathbf{district\_of\_hh} \doteq \exists\, \mathsf{has\_area} . \mathsf{g\_inside}_{\mathsf{p_2}} \sqcap \exists\, \mathsf{has\_area} . \neg\mathsf{equal}_{\mathsf{p_2}}$$

In the concept northern_german_region existential quantification for has_area is used to constrain the filler to be any polygon that is g_inside of p_5 which defines the area of Northern Germany (see Figure 2). In other words: The concept denoted by $\exists\, \mathsf{has\_area} . \mathsf{g\_inside}_{\mathsf{p_5}}$ subsumes every region of Northern Germany whose associated polygon is g_inside of p_5. Therefore, district_of_hh is automatically classified as a subconcept of northern_german_region. By analogy, we define the concepts for the federal states Hamburg and Schleswig-Holstein. We would like to emphasize that both concepts are subsumed by the concept northern_german_region.

$$\mathbf{federal\_state\_hh} \doteq \mathsf{german\_federal\_state} \sqcap \exists\, \mathsf{has\_area} . \mathsf{equal}_{\mathsf{p_2}}$$
$$\mathbf{federal\_state\_sh} \doteq \mathsf{german\_federal\_state} \sqcap \exists\, \mathsf{has\_area} . \mathsf{equal}_{\mathsf{p_4}}$$

These simple examples can already be represented with $\mathcal{ALC}(\mathcal{D})$. However, in many cases, restrictions about spatial relations have to be combined with additional conceptual restrictions. Thus, there is a need to quantify over roles that are defined based on predicates over a concrete domain. For example, how can we define a concept that describes a district of Hamburg that touches the "Federal State Hamburg" from the inside? Note that it is not sufficient that the corresponding district polygon (e.g. p_3 in Figure 2) is inside any polygon that is equal to the state polygon (e.g. p_2). The domain object that refers to the polygon p_2 with the feature has_area must also be subsumed by the concept federal_state_hh. We can adequately express this restriction with the help of the *role-forming predicate restriction* $(\exists (f)(f) . P)$. For modeling spatial relations we use role axioms for declaring corresponding roles in the TBox. The following TBox axioms also fulfill the restrictedness criterion because the nested concept terms employ only the $\exists f . P$ constructor.

$$\mathbf{is\_t\_inside} \doteq \exists (\mathsf{has\_area})(\mathsf{has\_area}) . \mathsf{t\_inside}$$
$$\mathbf{hh\_border\_district} \doteq \mathsf{district\_of\_hh} \sqcap \exists\, \mathsf{is\_t\_inside} . \mathsf{federal\_state\_hh}$$

The concept hh_border_district is discussed as an example for the use of the role-forming predicate restriction introduced by is_t_inside. The associated polygon of any individual that is a member this concept has to be in the t_inside relationship with another polygon that, in turn, is referred to by an instance of the concept federal_state_hh.

While the subsumption relationships discussed above are quite obvious, the advantages of TBox reasoning with spatial relations become apparent if we assume that the following axiom is computed by other components and added to our TBox (e.g. imagine a scenario employing machine learning techniques). The restrictedness criterion is fulfilled.

$$\mathbf{spatially\_related} \doteq \exists (\mathsf{has\_area})(\mathsf{has\_area}) . \mathsf{spatially\_related}$$
$$\mathbf{is\_touching} \doteq \exists (\mathsf{has\_area})(\mathsf{has\_area}) . \mathsf{touching}$$
$$\mathbf{unknown} \doteq \mathsf{district\_of\_hh} \sqcap \exists\, \mathsf{spatially\_related} . \mathsf{federal\_state\_hh} \sqcap \exists\, \mathsf{touching} . \mathsf{federal\_state\_sh}$$

If the polygon of district_of_hh touches the polygon of federal_state_sh, then the polygon of district_of_hh is also t_inside the polygon of federal_state_hh. Therefore, it can be proven that unknown is subsumed by hh_border_district (see the next section). The spatial constellation defined by the concept unknown could also be characterized as a "Hamburg border district to Schleswig-Holstein."

If district_of_hh had been defined without the term $\exists$ has_area . $\neg$equal$_{p_2}$, unknown would *not* have been subsumed by hh_border_district because an abstract individual whose associated polygon had been *equal* to p_2 would have been a member of unknown but not a member of hh_border_district (for hh_border_district the equal relation is eliminated by the constraint satisfaction process). A more detailed discussion of this example and the interaction between $\mathcal{ALCRP}(\mathcal{D})$ and the domain $\mathcal{D}_\mathcal{P}$ can be found in [18].

4 Conclusion

Based on the description logic $\mathcal{ALCRP}(\mathcal{D})$, we have shown how spatial and terminological reasoning can be combined in the TBox. Thus, the fruitful research on description logics has been extended to cope with qualitative spatial relations and quantitative spatial data. One of the main ideas is to introduce the notion of a role which is defined based on properties of concrete objects. The abstract domain is used to represent terminological knowledge about spatial objects on an abstract logical level. The concrete domain (space domain) extends the abstract domain and provides access to spatial reasoning algorithms. If required, even quantitative data (conrete polgyons) are considered by applying algorithms known from computational geometry. Techniques for spatial indexing can easily be integrated.

We admit that the $\mathcal{ALCRP}(\mathcal{D})$ restrictedness criterion for terminologies does impose tight constraints on modeling spatioterminological structures. However, in a specific application, many interesting concepts can be represented in a TBox with the additional advantage of having a decidable satisfiability algorithm. Our approach for testing satisfiability of finite conjunctions relies on current work in qualitative spatial reasoning theory [24]. Considering the topological spatial relations, it becomes clear that another instance of $\mathcal{ALCRP}(\mathcal{D})$ can deal with temporal relations. Similar constraint satisfaction algorithms known from the literature (e.g. [15]) can be employed. Future work will reveal the relationship between $\mathcal{ALCRP}(\mathcal{D})$ and, for instance, the temporal description logic developed in [1]. Although this approach does not impose restrictions on terminologies, it does not provide facilities for expressing value restrictions over spatial relations. This is not a problem in $\mathcal{ALCRP}(\mathcal{D})$ as long as the terminology fulfills the restrictedness criterion. Defined qualitative relations that are "grounded" in quantitative data provide a bridge to conceptual knowledge and support more extensive reasoning services to be exploited for solving practical problems.

Acknowledgments

We would like to thank Franz Baader who provided some of the ideas behind the PCP-reduction. Furthermore, we acknowledge the work of the anonymous reviewers who gave fruitful comments on this paper.

References

[1] A. Artale and E. Franconi. A temporal description logic for reasoning about actions and plans. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Fourth International Conference on Principles of Knowledge Representation*, May 1994.

[2] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *Twelfth International Joint Conference on Artificial Intelligence, 1991*, pages 452–457, August 1991.

[3] F. Baader and P. Hanschke. Extensions of concept languages for a mechanical engineering application. In H.J. Ohlbach, editor, *16th German Conference on Artificial Intelligence*, pages 132–143, September 1992.

[4] B. Bennett. Modal logics for qualitative spatial reasoning. *Bull. of the IGPL*, 3:1–22, 1995.

[5] S. Borgo, N. Guarino, and C. Masolo. A pointless theory of space based on strong connection and congruence. In L.C. Aiello, J. Doyle, and S. Shapiro, editors, *Fifth International Conference on Principles of Knowledge Representation*, pages 220–229, November 1996.

[6] A.G. Cohn, B. Bennett, J.M. Gooday, and N.M. Gotts. Representing and reasoning with qualitative spatial relations. In Stock [27], pages 97–134.

[7] G. De Giacomo and M. Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proceedings of the AAAI'94*, pages 205-212. AAAI Press / MIT Press, 1994.

[8] M.J. Egenhofer. Reasoning about binary topological relations. In O. Günther and H.-J. Schek, editors, *Advances in Spatial Databases, Second Symposium, SSD'91, Zurich, Aug. 28-30, 1991*, volume 525 of *LNCS*, pages 143–160. Springer-Verlag, August 1991.

[9] M. Grigni, D. Papadias, and C. Papadimitriou. Topological inference. In C. Mellish, editor, *Fourteenth International Joint Conference on Artificial Intelligence, 1995*, pages 901–906, August 1995.

[10] V. Haarslev. Formal semantics of visual languages using spatial reasoning. In *1995 IEEE Symposium on Visual Languages, Darmstadt, Germany, Sep. 5-9*, pages 156–163. IEEE Computer Society Press, Los Alamitos, September 1995.

[11] V. Haarslev and R. Möller. SBox: A qualitative spatial reasoner—progress report. In L. Ironi, editor, *11th International Workshop on Qualitative Reasoning, Pubblicazioni N. 1036, Istituto di Analisi Numerica C.N.R. Pavia*, pages 105–113, June 1997.

[12] V. Haarslev and R. Möller. Spatioterminological reasoning: Subsumption based on geometrical inferences. In Rousset et al. [25], pages 74–78.

[13] V. Haarslev, R. Möller, and C. Schröder. Combining spatial and terminological reasoning. In B. Nebel and L. Dreschler-Fischer, editors, *KI-94: Advances in Artificial Intelligence - Proc. 18th German Annual Conference on Artificial Intelligence*, volume 861 of *LNAI*, pages 142–153. Springer-Verlag, September 1994.

[14] V. Haarslev and M. Wessel. GenEd—an editor with generic semantics for formal reasoning about visual notations. In *1996 IEEE Symposium on Visual Languages, Boulder, Colorado, USA, Sep. 3-6*, pages 204–211. IEEE Computer Society Press, Los Alamitos, September 1996.

[15] P.B. Ladkin and A. Reinefeld. Fast algebraic methods for interval constraint problems. *Annals of Mathematics and Artificial Intelligence*, 19(3-4):383–411, 1997.

[16] C. Lutz, V. Haarslev, and R. Möller. A concept language with role-forming predicate restrictions. Technical Report FBI-HH-M-276/97, University of Hamburg, Computer Science Department, 1997.

[17] C. Lutz and R. Möller. Defined topological relations in description logics. In Rousset et al. [25], pages 15–19.

[18] R. Möller, V. Haarslev, and C. Lutz. Spatioterminological reasoning based on geometric inferences: The $\mathcal{ALCRP}(\mathcal{D})$ approach. Technical Report FBI-HH-M-277/97, University of Hamburg, Computer Science Department, 1997.

[19] B. Nebel. Computational properties of qualitative spatial reasoning: First results.. In I. Wachsmuth, C.-R. Rollinger, and W. Brauer, editors, *Proceedings, KI-95: Advances in Artificial Intelligence, 19th Annual German Conference on Artificial Intelligence*, volume 981 of *LNAI*, pages 233–244. Springer-Verlag, September 1995.

[20] E.M. Post. A variant of a recursively unsolvable problem. *Bull. Am. Math. Soc.*, 52:264–268, 1946.

[21] I. Pratt and O. Lemon. Ontologies for plane, polygonal mereotopology. *Notre Dame Journal of Formal Logic*, 2(38):225–245, 1997.

[22] D.A. Randell, Z. Cui, and A.G. Cohn. A spatial logic based on regions and connections. In B. Nebel, C. Rich, and W. Swartout, editors, *Principles of Knowledge Representation and Reasoning*, pages 165–176, October 1992.

[23] J. Renz. A canonical model of the region connection calculus. In T. Cohn and L. Schubert, editors, *Sixth International Conference on Principles of Knowledge Representation*, June 1998.

[24] J. Renz and B. Nebel. On the complexity of qualitative spatial reasoning: A maximal tractable fragment of the region connection calculus. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97)*, pages 522–527, August 1997.

[25] M.-C. Rousset et al., editor. *Proceedings of the International Workshop on Description Logics, DL'97, Sep. 27-29, 1997, Gif sur Yvette, France.* Universite Paris-Sud, Paris, September 1997.

[26] K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Twelfth International Conference on Artificial Intelligence*, pages 466–471, August 1991.

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

A model for reasoning about bidimensional temporal relations

Philippe Balbiani* Jean-François Condotta† Luis Fariñas del Cerro†

Abstract

This paper introduces the rectangle algebra as the power set of the set of the pairs of atomic relations which can hold between two rational intervals. It goes on proving that the question of the consistency of a rectangle network which constraints are preconvex is decidable by means of the path-consistency method in time polynomial in the length of the network.

1 Introduction

Temporal and spatial dependencies between data constitute the reason for existence of several problems in computer science : geographical information systems, natural language understanding, specification and verification of programs and systems, temporal and spatial databases, temporal and spatial planification, etc. Those who tackled these problems proposed numerous models for reasoning about time and space, the objects they considered as well as the relations between these objects distinguishing one of these models from the others. As an illustrative example, the model of the intervals proposed by Allen [1] is without doubt the better known model for reasoning about time whereas the model of the regions proposed by Cohn, Cui, Randell [7] is surely the best known model for reasoning about space.

On the interval's side, the objects are the rational intervals. Allen, Hayes [2], van Benthem [3] and Ladkin [8] introduced the first-order theory of intervals and proved that this theory is decidable and possesses the quantifier elimination property. Allen [1] introduced the interval network as a set of relational constraints between a finite number of interval variables and proposed a path-consistency algorithm for deciding the consistency of these interval networks. Ligozat [9] and Nebel, Bürckert [10] proved that the question of the consistency of an interval network which constraints are preconvex is decidable by means of the path-consistency method in time polynomial in the length of the network.

On the region's side, the objects are the regions of a topological space. Clark [5] [6] and Cohn, Cui, Randell [7] [11] introduced the first-order theory of regions but it is not known if this theory is decidable or possesses the quantifier elimination property. Nebel [12] introduced the region network as a set of relational constraints between a finite number of region variables and proved that the question of the consistency of a particular class of region networks is decidable in time polynomial in the length of the network.

The possible applications of the previous theories — interval algebra and region algebra — are numerous. However, we have no choice but to observe that : the interval algebra is about time and has nothing to say about space whereas simple relations like "x is on the left of y " or "x is below y " cannot be defined in the region algebra. That is the reason why this paper introduces a model for reasoning about bidimensional temporal relations. The objects of this model are the rational rectangles which sides are parallel to the axes of some orthogonal basis in a 2-dimensional Euclidean space. The relations between these objects are the 13×13 pairs of atomic relations which can hold between two rational intervals.

In the section 2, this paper presents the first-order theory of rational rectangles and relates it with the first-

*balbiani@lipn.univ-paris13.fr, Laboratoire d'informatique de Paris-Nord, Institut Galilée, Université Paris-Nord, Avenue Jean-Baptiste Clément, F-93430 Villetaneuse.

†condotta@irit.fr, farinas@irit.fr, Institut de recherche en informatique de Toulouse, Université Paul Sabatier, 118 route de Narbonne, F-31062 Toulouse Cedex.

order theory of rational intervals. In the sections 3, 4 and 5, it introduces the rectangle algebra as the power set of the set of the pairs of atomic relations which can hold between two rational intervals. In the section 5, it goes on proving that the question of the consistency of a rectangle network which constraints are preconvex is decidable by means of the path-consistency method in time polynomial in the length of the network.

2 Models

This section is devoted to the semantical analysis of the relationship between rational intervals and rational rectangles.

2.1 Interval models

According to Allen, Hayes [2] and Ladkin [8], an interval model is a relational structure of the form $\mathcal{M} = (\mathcal{I}, m)$ where $\mathcal{I}$ is a nonempty set of "intervals" and m is a binary relation on $\mathcal{I}$ such that :

- $m \circ m^{-1} \circ m \subseteq m$.
- $m \circ m = m \circ m \circ m$.
- $m \cap m^{-1} = \emptyset$.
- m and m^{-1} are serial.
- $m \cup m \circ m \cup m \circ m^{-1} \circ m^{-1} \circ m$ is the universal relation on $\mathcal{I}$.

The meaning of $x\ m\ y$ is "the right end of x and the left end of y are equal". Let $\equiv\ = (m^{-1} \circ m) \cap (m \circ m^{-1})$. The meaning of $x \equiv y$ is "the ends of x and y are equal". The reader may easily verify that : $\equiv$ is reflexive, symmetrical and transitive. $\mathcal{M}$ is normal when $\equiv$ is the identity relation on $\mathcal{I}$. Remark that $Int(\mathcal{Q})$ — the set of the rational intervals — is a normal interval model. The set of the interval models is a category which morphisms are the usual homomorphisms between relational structures : given two interval models $\mathcal{M} = (\mathcal{I}, m)$ and $\mathcal{M}' = (\mathcal{I}', m')$, a morphism of $\mathcal{M}$ to $\mathcal{M}'$ is a mapping h of $\mathcal{I}$ to $\mathcal{I}'$ such that, for every $x, y \in \mathcal{I}$, if $x\ m\ y$ then $h(x)\ m'\ h(y)$ and h is an isomorphism when h is bijective and such that if $h(x)\ m'\ h(y)$ then $x\ m\ y$.

2.2 Rectangle models

A rectangle model is a relational structure of the form $\mathcal{M} = (\mathcal{R}, m_1, m_2)$ where $\mathcal{R}$ is a nonempty set of "rectangles" and, for every $i \in \{1,2\}$, m_i is a binary relation on $\mathcal{R}$ such that $(\mathcal{R}, m_i)$ is an interval model. The meaning of $x\ m_1\ y$ (respectively : $x\ m_2\ y$) is "the right side of x and the left side of y are on the same vertical line" (respectively : "the top side of x and the bottom side of y are on the same horizontal line"), see fig. 1. For every $i \in \{1,2\}$, let $\equiv_i\ = (m_i^{-1} \circ m_i) \cap (m_i \circ m_i^{-1})$. The meaning of $x \equiv_1 \cap \equiv_2 y$ is "the vertices of x and y are equal". It is easy to verify the following conclusion, for every $i \in \{1,2\}$: $\equiv_i$ is reflexive, symmetrical and transitive. For every $i \in \{1,2\}$ and for every $x \in \mathcal{R}$, let $\equiv_i (x) = \{y : x \equiv_i y\}$. For every $i \in \{1,2\}$, let $\mathcal{R}_{\equiv_i} = \{\equiv_i (x) : x \in \mathcal{R}\}$. $\mathcal{M}$ is normal when $\equiv_1 \cap \equiv_2$ is the identity relation on $\mathcal{R}$ and $\equiv_1 \circ \equiv_2$ is the universal relation on $\mathcal{R}$. Observe that $Rec(\mathcal{Q})$ — the set of the rational rectangles — is a normal rectangle model. The set of the rectangle models is a category which morphisms are the usual homomorphisms between relational structures : given two rectangle models $\mathcal{M} = (\mathcal{R}, m_1, m_2)$ and $\mathcal{M}' = (\mathcal{R}', m_1', m_2')$, a morphism of $\mathcal{M}$ to $\mathcal{M}'$ is a mapping h of $\mathcal{R}$ to $\mathcal{R}'$ such that, for every $i \in \{1,2\}$ and for every $x, y \in \mathcal{R}$, if $x\ m_i\ y$ then $h(x)\ m_i'\ h(y)$ and h is an isomorphism when h is bijective and such that if $h(x)\ m_i'\ h(y)$ then $x\ m_i\ y$.

2.3 From intervals to rectangles

Let $\mathcal{M}_1 = (\mathcal{I}_1, m_1)$ and $\mathcal{M}_2 = (\mathcal{I}_2, m_2)$ be interval models. Let $\mathcal{R} = \mathcal{I}_1 \times \mathcal{I}_2$ and, for every $i \in \{1,2\}$, m_i' be the binary relation on $\mathcal{R}$ defined in the following way : $(x_1, x_2)\ m_i'\ (y_1, y_2)$ iff $x_i\ m_i\ y_i$. The reader may easily verify that :

Proposition 1 *$(\mathcal{R}, m_1', m_2')$ is a rectangle model.*

$(\mathcal{R}, m_1', m_2')$ is called rectangle model on $\mathcal{M}_1$ and $\mathcal{M}_2$. It is easy to verify that, for every $i \in \{1,2\}$, $(x_1, x_2) \equiv_i' (y_1, y_2)$ iff $x_i \equiv_i y_i$. All this goes to show that : If $\mathcal{M}_1$ is normal and $\mathcal{M}_2$ is normal then $(\mathcal{R}, m_1', m_2')$ is normal.

2.4 From rectangles to intervals

Let $\mathcal{M} = (\mathcal{R}, m_1, m_2)$ be a rectangle model. For every $i \in \{1,2\}$, let $\mathcal{I}_i = \mathcal{R}_{|\equiv_i}$ and m_i' be the binary relation on $\mathcal{I}_i$ defined in the following way : $\equiv_i (x)\ m_i' \equiv_i (y)$ iff $x\ m_i\ y$. It is easy to verify that, for every $i \in \{1,2\}$:

Proposition 2 *$(\mathcal{I}_i, m_i')$ is an interval model.*

For every $i \in \{1,2\}$, $(\mathcal{I}_i, m_i')$ is called interval model on $\mathcal{M}$ with respect to i. The reader may easily verify that, for every $i \in \{1,2\}$, $\equiv_i (x) \equiv_i' \equiv_i (y)$ iff $x \equiv_i y$. From all this it follows that, for every $i \in \{1,2\}$: $(\mathcal{I}_i, m_i')$ is normal.

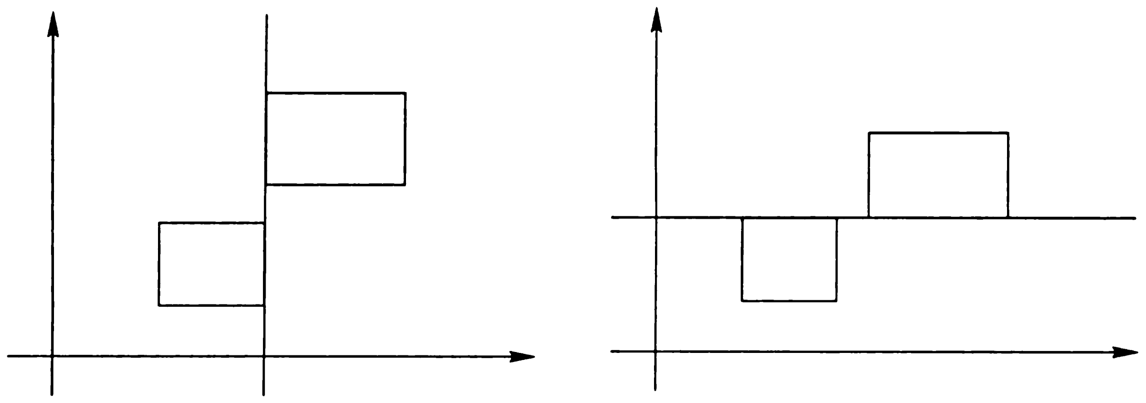


Figure 1: m_1 and m_2

2.5 Isomorphism

Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be interval models, $Rec(\mathcal{M}_1, \mathcal{M}_2)$ be the rectangle model on $\mathcal{M}_1$ and $\mathcal{M}_2$ and, for every $i \in \{1,2\}$, $\mathcal{M}'_i$ be the interval model on $Rec(\mathcal{M}_1, \mathcal{M}_2)$ with respect to i. We omit the elementary proof of the following conclusion, for every $i \in \{1,2\}$:

Proposition 3 *If $\mathcal{M}_i$ is normal then $\mathcal{M}_i$ and $\mathcal{M}'_i$ are isomorphic.*

Let $\mathcal{M}$ be a rectangle model, for every $i \in \{1,2\}$, $Int_i(\mathcal{M})$ be the interval model on $\mathcal{M}$ with respect to i and $\mathcal{M}'$ be the rectangle model on $Int_1(\mathcal{M})$ and $Int_2(\mathcal{M})$. The reader may easily verify that :

Proposition 4 *If $\mathcal{M}$ is normal then $\mathcal{M}$ and $\mathcal{M}'$ are isomorphic.*

Incidentally, we must not forget that, for every countable interval model $\mathcal{M}$: If $\mathcal{M}$ is normal then $\mathcal{M}$ and $Int(Q)$ are isomorphic. All this goes to show that, for every countable rectangle model $\mathcal{M}$: If $\mathcal{M}$ is normal then $\mathcal{M}$ and $Rec(Q)$ are isomorphic.

3 Algebras

This section introduces the rectangle algebra as the power set of the set of the pairs of the atomic relations which can hold between two rational intervals.

3.1 Interval algebra

3.1.1 Atomic relations

According to Allen [1], interval algebra has 13 atomic relations :

$$\mathcal{A}_{int} = \{p, m, o, s, d, f, pi, mi, oi, si, di, fi, eq\}$$

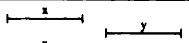
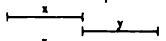
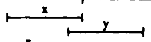
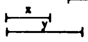
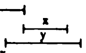
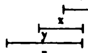
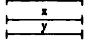
Relation	Symbol	Reverse	Meaning
precedes	p	pi	
meets	m	mi	
overlaps	o	oi	
starts	s	si	
during	d	di	
finishes	f	fi	
equals	eq	eq	

Figure 2: Atomic relations in the interval algebra

which constitute the exhaustive list of the relations which can hold between two rational intervals, see fig. 2. According to Bessière, Isli, Ligozat [4] and Ligozat [9], these atomic relations are arranged in ascending order and this ascending order defines a lattice called interval lattice (see fig. 3) whereas the interval algebra is defined as the power set of $\mathcal{A}_{int}$. The atomic relations of $\mathcal{A}_{int}$ will be denoted by the letters A, B, C, D whereas the sets of such atomic relations will be denoted by the letters R, S, T, U.

Allen and Hayes [2] have observed that the 13 atomic relations of the interval algebra are definable by means of the binary relation m together with the fundamental operations of intersection, composition and reverse.

Example 1 *The atomic relation p corresponds to the relational expression $m \circ m$ while the atomic relation d corresponds to the relational expression $(mi \circ mi \circ m) \cap (m \circ m \circ mi)$.*

3.1.2 Convex relations

Convex relations are those relations in $2^{\mathcal{A}_{int}}$ which correspond to intervals in the interval lattice.

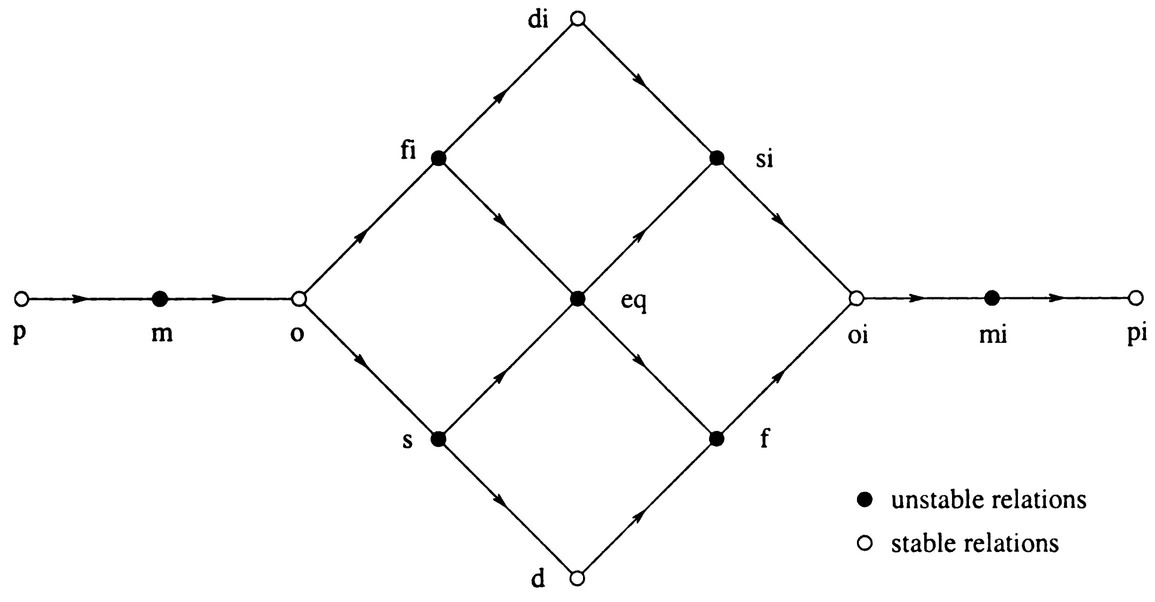


Figure 3: The interval lattice

Example 2 *The convex relation $\{m, o, s, fi, eq\}$ corresponds to the interval $[m, eq]$.*

It is easy to verify that the class of convex relations in $2^{\mathcal{A}_{int}}$ is stable for intersection.

3.1.3 Preconvex relations

According to Ligozat [9], unstable relations are those atomic relations in $\mathcal{A}_{int}$ which correspond to black circles in the representation of the interval lattice. The preconvex relations in $2^{\mathcal{A}_{int}}$ are defined by induction in the following way :

- For every convex relation R in $2^{\mathcal{A}_{int}}$, R is a preconvex relation in $2^{\mathcal{A}_{int}}$.
- For every preconvex relation R in $2^{\mathcal{A}_{int}}$ and for every unstable relation A in $\mathcal{A}_{int}$, $R \setminus \{A\}$ is a preconvex relation in $2^{\mathcal{A}_{int}}$.

Example 3 *The preconvex relation $\{p, o, s, si, fi, di\}$ is obtained from $[p, si]$ by removing the unstable relations m and eq.*

As well, the reader may easily verify that the class of preconvex relations in $2^{\mathcal{A}_{int}}$ is stable for intersection.

3.2 Rectangle algebra

3.2.1 Atomic relations

Rectangle algebra has 13×13 atomic relations : $\mathcal{A}_{rec} = \{(A, B) : A, B \in \mathcal{A}_{int}\}$ which constitute the exhaustive list of the relations which can hold between two rational rectangles. These atomic relations are arranged in ascending order in the following way : $(A, B) \preceq (C, D)$ iff $A \leq C$ and $B \leq D$. This ascending order defines a lattice called rectangle lattice whereas the rectangle algebra is defined as the power set of $\mathcal{A}_{rec}$. The sets of such atomic relations will be denoted by the capital letters R, S.

The reader may easily verify that the 13×13 atomic relations of the rectangle algebra are definable by means of the binary relations (m, eq) and (eq, m) together with the fundamental operations of intersection, composition and reverse.

Example 4 *The atomic relation (p, d) corresponds to the relational expression $((m, eq) \circ (m, eq)) \circ (((eq, mi) \circ (eq, mi) \circ (eq, m)) \cap ((eq, m) \circ (eq, m) \circ (eq, mi)))$.*

3.2.2 Saturated relations

Saturated relations are those relations in $2^{\mathcal{A}_{rec}}$ which are obtained through the Cartesian product of two relations in $2^{\mathcal{A}_{int}}$.

Example 5 *The saturated relation $\{(p, m), (p, s), (p, d), (o, m), (o, s), (o, d)\}$ corresponds to the Cartesian product $\{p, o\} \times \{m, s, d\}$.*

It is easy to verify that, for every relation R, S, T, U in $2^{\mathcal{A}_{int}}$: $(R \times S) \cap (T \times U) = (R \cap T) \times (S \cap U)$. Consequently, the class of saturated relations in $2^{\mathcal{A}_{rec}}$ is stable for intersection.

3.2.3 Convex relations

Like in $2^{\mathcal{A}_{int}}$, convex relations are those relations in $2^{\mathcal{A}_{rec}}$ which correspond to intervals in the rectangle lattice.

Example 6 $\{(p,s),(p,d),(p,f),(p,eq),(m,s),(m,d),(m,f),(m,eq)\}$ *is a convex relation that corresponds to the interval* $[(p,s),(m,f)]$. *The convex relation "the right side of x is on the left of the left side of y" corresponds to the interval* $[(p,p),(p,pi)]$ *whereas the convex relation "the top side of x is below the bottom side of y" corresponds to the interval* $[(p,p),(pi,p)]$.

We omit the elementary proof of the following conclusion, for every atomic relation A, B, C, D in $\mathcal{A}_{int}$, if $A \leq C$ and $B \leq D$ then :

Proposition 5 $[(A,B),(C,D)] = [A,C] \times [B,D]$.

Consequently :

Proposition 6 *The class of convex relations in* $2^{\mathcal{A}_{rec}}$ *coincides with the class of the relations which are obtained through the Cartesian product of two convex relations in* $2^{\mathcal{A}_{int}}$. *Moreover, every convex relation in* $2^{\mathcal{A}_{rec}}$ *is saturated.*

All this goes to show that the class of convex relations in $2^{\mathcal{A}_{rec}}$ is stable for intersection.

3.2.4 Preconvex relations

The preconvex relations in $2^{\mathcal{A}_{rec}}$ are defined by induction in the following way :

- For every convex relation R in $2^{\mathcal{A}_{rec}}$, R is a preconvex relation in $2^{\mathcal{A}_{rec}}$.
- For every preconvex relation R in $2^{\mathcal{A}_{rec}}$ and for every unstable relation A in $\mathcal{A}_{int}$, $R \backslash (\{A\} \times \mathcal{A}_{int})$ is a preconvex relation in $2^{\mathcal{A}_{rec}}$.
- For every preconvex relation R in $2^{\mathcal{A}_{rec}}$ and for every unstable relation B in $\mathcal{A}_{int}$, $R \backslash (\mathcal{A}_{int} \times \{B\})$ is a preconvex relation in $2^{\mathcal{A}_{rec}}$.

Example 7
$\{(p,o),(p,d),(m,o),(m,d),(o,o),(o,d)\}$ *is a preconvex relation that is obtained from* $[(p,m),(s,d)]$ *by removing the relations* (s,m), (s,o), (s,s), (s,d), (p,m), (m,m), (o,m), (p,s), (m,s), (o,s).

It may be asserted that, for every preconvex relation R, S in $2^{\mathcal{A}_{int}}$ and for every unstable relation A, B in $\mathcal{A}_{int}$:

Proposition 7 $(R \times S) \backslash (\{A\} \times \mathcal{A}_{int}) = (R \backslash \{A\}) \times S$ *and* $(R \times S) \setminus (\mathcal{A}_{int} \times \{B\}) = R \times (S \setminus \{B\})$.

Consequently :

Proposition 8 *The class of preconvex relations in* $2^{\mathcal{A}_{rec}}$ *coincides with the class of the relations which are obtained through the Cartesian product of two preconvex relations in* $2^{\mathcal{A}_{int}}$. *Moreover, every preconvex relation in* $2^{\mathcal{A}_{rec}}$ *is saturated.*

From all this it follows that the class of preconvex relations in $2^{\mathcal{A}_{rec}}$ is stable for intersection.

4 Composition

This section introduces the composition of two relations in $2^{\mathcal{A}_{rec}}$ on the basis of the composition of two relations in $2^{\mathcal{A}_{int}}$.

4.1 Interval composition

Composition between atomic relations in $\mathcal{A}_{int}$ is defined through the table of composition of Allen [1].

Example 8 $d \circ p = \{p\}$ *and* $o \circ mi = \{oi, si, di\}$.

Composition between relations in $2^{\mathcal{A}_{int}}$ is defined in the following way : $R \circ S = \bigcup\{A \circ B : A \in R$ and $B \in S\}$. Incidentally, we must not forget that the class of convex relations in $2^{\mathcal{A}_{int}}$ and the class of preconvex relations in $2^{\mathcal{A}_{int}}$ are stable for composition.

4.2 Rectangle composition

Composition between atomic relations in $\mathcal{A}_{rec}$ is defined in the following way : $(A,B) \circ (C,D) = (A \circ C) \times (B \circ D)$.

Example 9 $(d,o) \circ (p,mi) = \{p\} \times \{oi,si,di\} = \{(p,oi),(p,si),(p,di)\}$.

Composition between relations in $2^{\mathcal{A}_{rec}}$ is defined in the following way : $R \circ S = \bigcup\{(A \circ C) \times (B \circ D) : (A,B) \in R$ and $(C,D) \in S\}$. We omit the elementary proof of the following conclusion, for every relation R, S, T, U in $2^{\mathcal{A}_{int}}$:

Proposition 9 $(R \times S) \circ (T \times U) = (R \circ T) \times (S \circ U)$.

Consequently, the class of convex relations in $2^{\mathcal{A}_{rec}}$ and the class of preconvex relations in $2^{\mathcal{A}_{rec}}$ are stable for composition.

5 Networks

This section introduces the rectangle network as a set of relational constraints between a finite number of rectangle variables and goes on proving that the question of the consistency of a rectangle network which constraints are preconvex is decidable by means of the path-consistency method in time polynomial in the length of the network.

5.1 Interval network

An interval network is a structure of the form (n, R) where $n \geq 1$ and R is a mapping of $(n) \times (n)$ to the set of the relations in $2^{\mathcal{A}_{int}}$ such that, for every $i \in (n)$, $R_{ii} = \{eq\}$ and, for every $i, j \in (n)$, R_{ji} is the converse of R_{ij}. Let (n, R) be an interval network :

- (n, R) is path-consistent when, for every $i, j \in (n)$, $R_{ij} \neq \emptyset$ and, for every $i, j, k \in (n)$, $R_{ij} \circ R_{jk} \supseteq R_{ik}$.
- (n, R) is consistent when there exists a mapping x of (n) to the set of the rational intervals such that, for every $i, j \in (n)$, there exists an atomic relation A in $\mathcal{A}_{int}$ such that $A \in R_{ij}$ and x_i and x_j satisfy A in the rational interval model.
- (n, R) is convex (respectively : preconvex) when, for every $i, j \in (n)$, R_{ij} is a convex (respectively : preconvex) relation in $2^{\mathcal{A}_{int}}$.

According to Allen [1], the path-consistency method — which consists in successively replacing, for every $i, j, k \in (n)$, the constraint R_{ik} by the constraint $(R_{ij} \circ R_{jk}) \cap R_{ik}$ — is not complete as a decision procedure for the issue of the consistency of an interval network. Incidentally, we must not forget that, according to Ligozat [9] and Nebel, Bürckert [10], for every interval network (n, R), if (n, R) is preconvex and path-consistent then (n, R) is consistent.

5.2 Rectangle network

A rectangle network is a structure of the form (n, R) where $n \geq 1$ and R is a mapping of $(n) \times (n)$ to the set of the relations in $2^{\mathcal{A}_{rec}}$ such that, for every $i \in (n)$, $R_{ii} = \{(eq, eq)\}$ and, for every $i, j \in (n)$, R_{ji} is the converse of R_{ij}. Let (n, R) be a rectangle network :

- (n, R) is saturated when, for every $i, j \in (n)$, R_{ij} is a saturated relation in $2^{\mathcal{A}_{rec}}$.
- (n, R) is path-consistent when, for every $i, j \in (n)$, $R_{ij} \neq \emptyset$ and, for every $i, j, k \in (n)$, $R_{ij} \circ R_{jk} \supseteq R_{ik}$.
- (n, R) is consistent when there exists a mapping x of (n) to the set of the rational rectangles such that, for every $i, j \in (n)$, there exists an atomic relation A in $\mathcal{A}_{int}$ and there exists an atomic relation B in $\mathcal{A}_{int}$ such that $(A, B) \in R_{ij}$ and x_i and x_j satisfy (A, B) in the rational rectangle model.
- (n, R) is convex (respectively : preconvex) when, for every $i, j \in (n)$, R_{ij} is a convex (respectively : preconvex) relation in $2^{\mathcal{A}_{rec}}$.

Like in $2^{\mathcal{A}_{int}}$, the path-consistency method is not complete as a decision procedure for the issue of the consistency of a rectangle network. If (n, R) is saturated then there exists an interval network (n, S) and there exists an interval network (n, T) such that, for every $i, j \in (n)$, $R_{ij} = S_{ij} \times T_{ij}$. Moreover :

- If (n, R) is path-consistent then, for every $i, j \in (n)$, $R_{ij} \neq \emptyset$ and, for every $i, j, k \in (n)$, $R_{ij} \circ R_{jk} \supseteq R_{ik}$. Consequently, for every $i, j \in (n)$, $S_{ij} \neq \emptyset$, $T_{ij} \neq \emptyset$ and, for every $i, j, k \in (n)$, $(S_{ij} \times T_{ij}) \circ (S_{jk} \times T_{jk}) \supseteq (S_{ik} \times T_{ik})$, $(S_{ij} \circ S_{jk}) \times (T_{ij} \circ T_{jk}) \supseteq (S_{ik} \times T_{ik})$ and $S_{ij} \circ S_{jk} \supseteq S_{ik}$ and $T_{ij} \circ T_{jk} \supseteq T_{ik}$. Consequently, (n, S) and (n, T) are path-consistent. Reciprocally, it is easy to verify that if (n, S) and (n, T) are path-consistent then (n, R) is path-consistent.
- If (n, R) is consistent then there exists a mapping x of (n) to the set of the rational rectangles such that, for every $i, j \in (n)$, there exists an atomic relation A in $\mathcal{A}_{int}$ and there exists an atomic relation B in $\mathcal{A}_{int}$ such that $(A, B) \in R_{ij}$ and x_i and x_j satisfy (A, B) in the rational rectangle model. Let y be the mapping of (n) to the set of the rational intervals such that, for every $k \in (n)$, y_k is the projection of x_k onto the horizontal axis and z be the mapping of (n) to the set of the rational intervals such that, for every $k \in (n)$, z_k is the projection of x_k onto the vertical axis. Consequently, for every $i, j \in (n)$, there exists an atomic relation A in $\mathcal{A}_{int}$ such that $A \in S_{ij}$ and y_i and y_j satisfy A in the rational interval model and there exists an atomic relation B in $\mathcal{A}_{int}$ such that $B \in T_{ij}$ and z_i and z_j satisfy B in the rational interval model. Consequently, (n, S) and (n, T) are consistent. Reciprocally, it is easy to verify that if (n, S) and (n, T) are consistent then (n, R) is consistent.

From all this it follows that, for every rectangle network (n, R) :

Proposition 10 *If (n, R) is saturated then there exists an interval network (n, S) and there exists an in-*

terval network (n,T) such that, for every $i,j \in (n)$, $R_{ij} = S_{ij} \times T_{ij}$. Moreover, (n,R) is path-consistent (respectively : consistent) iff (n,S) and (n,T) are path-consistent (respectively : consistent).

All this goes to show that, for every rectangle network (n,R) :

Proposition 11 *If (n,R) is preconvex and path-consistent then (n,R) is consistent.*

Consequently, the question of the consistency of a rectangle network which constraints are preconvex is decidable by means of the path-consistency method in time polynomial in the length of the network.

6 Conclusion

To close our study of $2^{\mathcal{A}_{rec}}$, we mention some questions that remain unsolved :

- The first-order theory of rational intervals is decidable and possesses the quantifier elimination property. Prove or disprove that the first-order theory of rational rectangles is decidable and possesses the quantifier elimination property.
- The class of preconvex relations in $2^{\mathcal{A}_{int}}$ is the maximal tractable class that contain all the 13 atomic relations in $\mathcal{A}_{int}$. Prove or disprove that the class of preconvex relations in $2^{\mathcal{A}_{rec}}$ is the maximal tractable class that contain all the 13×13 atomic relations in $\mathcal{A}_{rec}$.

Acknowledgement

This work was supported in part by CENA under project n° 94/C0018. Special acknowledgement is heartly granted to Gérard Ligozat who made several helpful comments for improving the readability of the paper.

References

[1] **J. Allen.** *Maintaining knowledge about temporal intervals.* Communications of the ACM, Volume 26, 832–843, 1983.

[2] **J. Allen, P. Hayes.** *A common sense theory of time.* A. Joshi (editor), IJCAI-85. Proceedings of the 9th International Joint Conference on Artificial Intelligence, Volume 1, 528–531, Morgan Kaufmann, 1985.

[3] **J. van Benthem.** *The Logic of Time.* Reidel, 1983.

[4] **C. Bessière, A. Isli, G. Ligozat.** *Global consistency in interval algebra networks : tractable subclasses.* W. Wahlster (editor), ECAI-96. Proceedings of the 12th European Conference on Artificial Intelligence, 3–7, Wiley, 1996.

[5] **B. Clark.** *A calculus of individuals based on "connection".* Notre Dame Journal of Formal Logic, Volume 22, 204–218, 1981.

[6] **B. Clark.** *Individuals and points.* Notre Dame Journal of Formal Logic, Volume 26, 61–75, 1985.

[7] **T. Cohn, Z. Cui, D. Randell.** *Taxonomies of logically defined qualitative spatial rules.* Journal of Human-Computer Studies, Volume 43, 831–846, 1995.

[8] **P. Ladkin.** *Models of axioms for time intervals.* AAAI-87. Proceedings of the 6th National Conference on Artificial Intelligence, Volume 1, 234–239, Morgan Kaufmann, 1987.

[9] **G. Ligozat.** *A new proof of tractability for ORD-Horn relations.* AAAI-96. Proceedings of the 13th National Conference on Artificial Intelligence, Volume One, 395–401, AAAI Press and MIT Press, 1996.

[10] **B. Nebel, H.-J. Bürckert.** *Reasoning about temporal relations : a maximal tractable subclass of Allen's interval algebra.* AAAI-94. Proceedings of the 12th National Conference on Artificial Intelligence, Volume One, 356–361, AAAI Press and MIT Press, 1994.

[11] **D. Randell, Z. Cui, T. Cohn.** *An interval logic for space based on "connection".* B. Neumann (editor), ECAI-92. Proceedings of the 10th European Conference on Artificial Intelligence, 394–398, Wiley, 1992.

[12] **J. Renz, B. Nebel.** *On the complexity of qualitative spatial reasoning : a maximal tractable fragment of the region connection calculus.* M. Pollack (editor), IJCAI-97. Proceedings of the 15th International Joint Conference on Artificial Intelligence, Volume 1, 522–527, Morgan Kaufmann, 1997.

A Qualitative Theory of Motion Based on Spatio-Temporal Primitives

Philippe Muller
IRIT
Université Paul Sabatier
118 route de Narbonne, 31062 Toulouse Cedex
muller@irit.fr

Abstract

This paper presents a formal theory for reasoning about motion of spatial entities, in a qualitative framework. Taking over a theory intended for spatial entities, we enrich it to achieve a theory whose intended models are spatio-temporal entities, an idea sometimes proposed by philosophers or AI authors but never fully exploited. We show what kind of properties usually assumed as desirable parts of any space-time theory are recovered from our model, thus giving a sound theoretical basis for a natural, qualitative representation of motion.

1 INTRODUCTION

This paper presents a theory for representing and reasoning about motion in a qualitative framework. A lot of work has been devoted to the representation of space in the past few years, as spatial concepts pervades many domains of AI. Part of these works have focused on the building of theories for reasoning about incomplete and/or imprecise information as such theories would be more easily exploited than the traditional quantitative models (widely used in robotics for instance). For that purpose, in what is called "Qualitative Spatial Reasoning" (QSR), a lot of models make use of regions of space as their ontological primitives rather than the points of traditional geometry (for a general presentation, see [Vieu, 1997]). This ontological shift imposes to build new theories for most spatial and geometrical concepts. Several studies have been devoted to the building of mereo-topological theories on that basis (for instance the well-known RCC theory [Randell *et al.*, 1992]; see also [Fleck, 1996],[Asher and Vieu, 1995] and for a survey of mereo-topology [Varzi, 1996]), as mereo-topology (essentially, the modeling of connection and part-hood) is regarded as the basis of common-sense geometry.

Very little work has been done on motion in such a qualitative framework. Even works done under the banner of so-called "qualitative physics" usually make use of the classical quantitative model of kinematics. Motion is thus usually represented in a Newtonian/Cartesian framework [Rajagopalan and Kuipers, 1994],[Hays, 1989]. Other studies focus more on related aspects, for instance [Forbus, 1983, Davis, 1989] insist more on the concept of dynamic process, and [Shanahan, 1995, Hartley, 1992] on default reasoning in metric spaces that are not really clearly characterized. Motion is nonetheless a key notion in our understanding of spatial relations; indeed, changes of state in RCC has been analyzed through transition graphs in which the relations form so-called "conceptual neighborhoods", via potential motion. Only certain changes are allowed, assuming continuous change between relations (this means spatial changes are restricted to the edges of such graphs). Continuity is the central notion here, but remains implicitly assumed without a formal definition; only the work of Galton [Galton, 1993, Galton, 1997] has begun to address what continuity implies for a common-sense theory of motion. Still, this kind of work characterizes continuity as a set of logical constraints on the transitions in a temporal framework and does not add much insight to the already existing transition graph (it is more descriptive than explanatory), and falls short of an explicit, generic characterization of spatio-temporal continuity.

We present here a mereo-topological model in which the primitive entities are spatio-temporal regions, on which we define both spatio-temporal and temporal relations. These entities can be interpreted as the trajectories of physical objects and events. Such an approach is not entirely new since Russell [Russell, 1914] or Carnap [Carnap, 1958] had already suggested it, for reasons analyzed in [Simons, 1987], and Hayes has begun to show the use that could be made of such objects for common-sense reasoning [Hayes, 1985b, Hayes, 1985a]; mainly, the homogeneity of a theory which treats events and objects at the same level has an interesting expressive power, and directly addresses the problem of objects identity through time. We show that a qualitative notion of continuity can be expressed in such a theory, which allows for a full characterization of conceptual neighborhoods of spatial relations. Section 2 presents the topological theory (of Asher & Vieu) we take as a basis, section 3 and 4 present our extension to that theory, along with the definition of continuity within the theory and a presentation of how we recover conceptual neighborhoods from this general definition. Section 5 shows how to define some common-sense concepts related to motion.

2 THE TOPOLOGICAL BASE

In this section we take over the theory of topological concepts presented in [Asher and Vieu, 1995], which is shown to be consistent and complete with respect to a certain class of models, and which has at least the same expressive power as RCC8 [Randell *et al.*, 1992]; both theories are based on the work of Clarke [Clarke, 1981]. The main difference between them is that RCC8 does not distinguish between open and closed regions whereas Asher and Vieu's theory does. It is not obvious that the results we present here could be demonstrated in RCC, because of this difference. Moreover Asher and Vieu's theory has been thought as a spatio-temporal theory from the beginning, although this aspect has never been much exploited by the authors (not even in a formal account of motion verbs in [Asher and Sablayrolles, 1995]). Objects of this theory can be interpreted as the spatio-temporal referents of physical objects or events, or regions of space. Figure 2 shows the classical intended 2D spatial interpretation of some relations that can be defined from the primitive relation C (connection), while Figure 1 shows an illustration of how a spatio-temporal relation is to be intuitively understood. Here the "spatial" dimension of an entity is a part of the horizontal axis while the temporal evolution of the entity is shown along the vertical axis. Space could of course be 2- or 3-dimensional, and this Figure is only one example of what our intended models are.

We will now present the topological axioms from [Asher and Vieu, 1995]. For the sake of clarity, in the following, universal quantifiers scoping over whole formulas are omitted. Upper case symbols stand for predicates, lower case ones for variables or constants. The symbol $\triangleq$ stands for definition. We use a first order language with equality.

A 2.1 Cxx

A 2.2 $Cxy \rightarrow Cyx$

A 2.3 $(\forall z\ (Czx \leftrightarrow Czy) \rightarrow x = y)$

A lot of relations can be defined on that sole basis, we will make use of the following (see [Asher and Vieu, 1995] for the formal definition): P (part of), PP, (proper part of), O

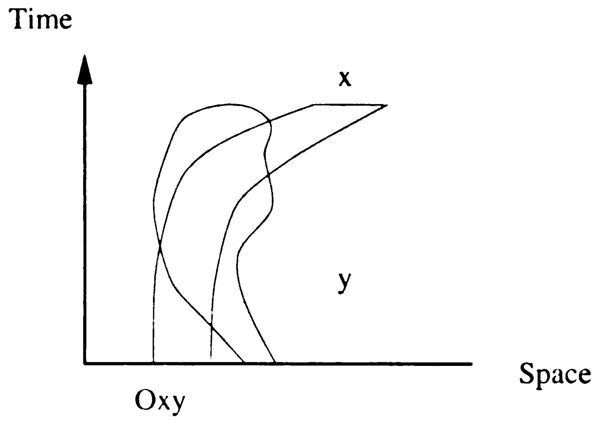


Figure 1: A Spatio-Temporal Interpretation of O(verlap)

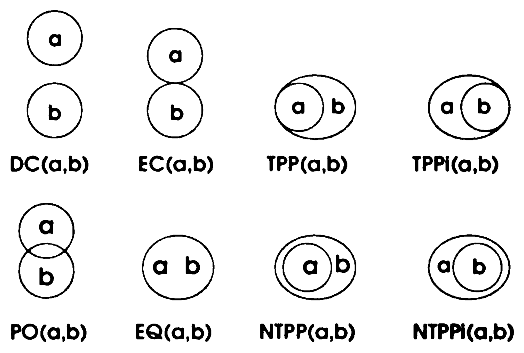


Figure 2: A 2D Spatial Interpretation of Topological Relations

(overlap), PO (partial overlap), EC (external connection), TP (tangential part), NTP (non tangential part), TPP and NTPP and the converse relations TPP^{-1} and $NTPP^{-1}$ (also written TPPI and NTPPI), cf Figure 2. The following are existence axioms of classical operators [Asher and Vieu, 1995]:

A 2.4 $\forall x \forall y \exists z\ \forall u (Cuz \leftrightarrow (Cux \vee Cuy))$
(sum, noted $x+y$)

A 2.5 $\forall x \exists y\ \neg Cyx \rightarrow \exists z\ \forall u (Cuz \leftrightarrow \exists v\ (\neg Cvx \wedge Cvu))$
(complement, noted $-x$)

A 2.6 $\exists x\ \forall u\ \ Cux$
(existence of a universe, noted a^*)

A 2.7 $Oxy \rightarrow \exists z\ \forall u (Cuz \leftrightarrow \exists v\ (Pvx \wedge Pvy \wedge Cvu))$
(intersection, noted $x \cdot y$)

A 2.8 $\forall x \exists y\ \forall u\ \ (Cuy \leftrightarrow \exists v\ (NTPvx \wedge Cvu))$
(interior, noted ix)

D 2.1 (closure): $cx \triangleq -i(-x)$

A 2.9 $c(a^*) = a^*$

The topological operators are constrained in a classical way:

D 2.2 $\mathrm{OP}x \triangleq (ix = x)$ (definition of an open)
A 2.10 $(\mathrm{OP}x \wedge \mathrm{OP}y \wedge \mathrm{O}xy) \rightarrow \mathrm{OP}(x \cdot y)$

The authors also define a few useful notions:

D 2.3 $\mathrm{SP}xy \triangleq \neg \mathrm{C}cxcy$ (separateness[Vieu, 1991])
D 2.4 $\mathrm{CON}x \triangleq \neg(\exists x_1 \exists x_2 (x = x_1 + x_2 \wedge \mathrm{SP}x_1 x_2))$ (self-connectedness)
D 2.5 $\mathrm{AT}x \triangleq \neg \exists y \mathrm{PP}yx$ (atomicity)

We define quasi-atoms, whose only proper parts are their interiors:
D 2.6 $\mathrm{QAT}x \triangleq \mathrm{AT}ix \wedge \neg \mathrm{OP}x$

and "generalized" atoms, which can be atoms or quasi-atoms:
D 2.7 $\mathrm{GAT}x \triangleq \mathrm{AT}x \vee \mathrm{QAT}x$

and are conceptually close to atoms (the only difference being that they can include part of the closure of an atom).

3 TEMPORAL RELATIONS

The mere topology we have introduced is quite general and does not imply a spatio-temporal interpretation; in order to do so it is necessary to capture a notion of temporal order between entities of the theory. The kind of properties we want are related to the assumptions already made about those entities: they are extended in a primitive space-time so that temporal relations actually bear on the spatio-temporal entities *themselves*, and not on time extents. Thus our temporal relations are close to event logics (see e.g. Kamp [Kamp, 1979]), which are quite similar to interval-based temporal logics (see vanBenthem [van Benthem, 1995] or Galton [Galton, 1995] for a general presentation), with the difference that two objects can be different and still be contemporaneous. However these logics are often based on two primitives, < for precedence, and either inclusion or overlap. If we want to keep the topological distinction between connection and overlap made on the spatio-temporal entities, we need to have topological concepts on the temporal level as well (and Varzi [Varzi, 1996] has shown this is not possible with only overlap or inclusion, i.e. mereology). This distinction is part of Allen's common-sense theory of time [Allen and Hayes, 1985] and Galton [Galton, 1993] has shown it is necessary to account for continuous motion. But Allen's theory assumes self-connected intervals and Galton assumes the existence of both instants and intervals. We believe however that none of these hypotheses are necessary and keep a more general model where entities can be disconnected; it is moreover an ontologically more parsimonious (assuming only extended entities) model which turns out to be quite practical. We thus introduce a relation of temporal connection $\between$ (interpreted in much the same way as a temporal 'C') besides the obvious <. For readability's sake, and to distinguish them from the spatio-temporal relations, temporal relations are infixed.

A 3.1 $x \between y \rightarrow y \between x$ (symmetry)

A 3.2 $x \between x$ (reflexivity)

A 3.3 $x \between y \rightarrow \neg x < y$
(incompatibility between $\between$ and <)

A 3.4 $x < y \rightarrow \neg y < x$ (antisymmetry of <)

A 3.5 $(x < y \wedge y \between z \wedge z < t) \rightarrow x < t$
(composition of $\between$ and <)

From this we can define the intuitive relations:

D 3.1 $x \subseteq_t y \triangleq \forall z (z \between x \rightarrow z \between y)$
(temporal inclusion)

D 3.2 $x \sigma y \triangleq \exists z (z \subseteq_t y \wedge z \subseteq_t x)$
(temporal overlap)

D 3.3 $(x \equiv_t y) \triangleq x \subseteq_t y \wedge y \subseteq_t x$
(temporal equivalence)

The transitivity of < can be derived from A3.2 and A3.5 and its irreflexivity from A3.3 and A3.2. The relation < is therefore a strict partial order. Figure 3 gives an illustration of the temporal relations between spatio-temporal entities.

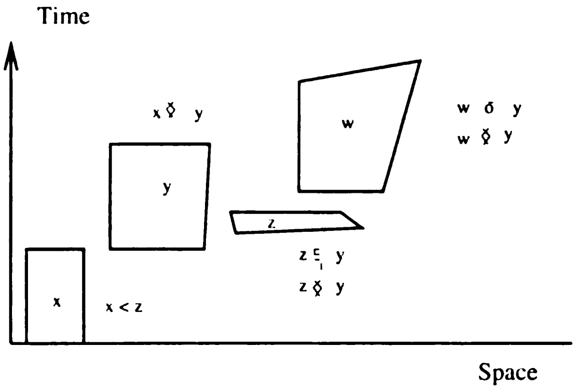


Figure 3: Temporal Relations Illustrated

We can derive all the axioms of other systems based on < and ($\subseteq_t$ or σ) (cf. [van Benthem, 1995, Aurnague and Vieu, 1993, Kamp, 1979]) from the previous axioms, which are much inspired by [Aurnague and Vieu, 1993]. We can also define a notion of temporal connectedness:

D 3.4
$\mathrm{CON}_t x \triangleq \neg(\exists x_1 \exists x_2 (x = x_1 + x_2 \ \wedge \ \neg(cx_1 \bowtie cx_2)))$

We can moreover impose the linearity of the underlying temporal order by stating that there must be a relation ($<$ or $\bowtie$) between two temporally self-connected entities (and only in that case since the theory allows for the sum of arbitrary entities, therefore not necessarily self-connected). The following axiom indeed eliminates branching time models.

A 3.6 $(\mathrm{CON}_t x \wedge \mathrm{CON}_t y) \rightarrow (x < y \ \vee \ x \bowtie y \ \vee \ y < x)$

4 SPATIO-TEMPORAL INTERACTIONS

4.1 LINKS BETWEEN TIME AND SPACE-TIME

The links between spatio-temporal relations and temporal relations have yet to be defined. Some axioms were defined in the same perspective in [Vieu, 1991], using a "temporal overlap"relation, but were not part of a larger motion theory[1].

A 4.1 $\mathrm{C}xy \rightarrow x \bowtie y$ (two connected entities are also time-connected)

The models must not be temporal only, so C and $\bowtie$ are different (4.2) and there is at least two temporally ordered entities (4.3).

A 4.2 $\exists x \exists y \ x \bowtie y \ \wedge \ \neg \mathrm{C}xy$

A 4.3 $\exists x \exists y \ x < y$

The interaction between + and $<$ and $\bowtie$ need to be specified:

A 4.4 $(x < y \wedge z < y) \leftrightarrow (x+z) < y$

A 4.5 $(x+y) \bowtie z \leftrightarrow \ x \bowtie z \ \vee \ y \bowtie z$

In order to define relations between parts of trajectories that can vary through time, we now define a notion of "temporal slice", i.e. the maximal part corresponding to a certain time extent (Figure 4).

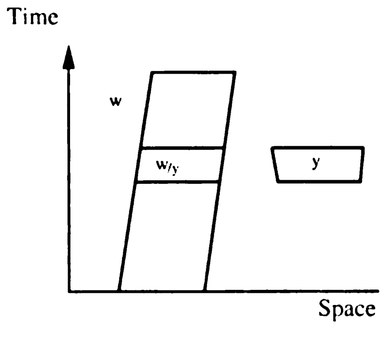


Figure 4: Illustration of a Temporal Slice

[1]The axioms of [Vieu, 1991] are theorems of our theory.

D 4.1 $\mathrm{TS}xy \triangleq \mathrm{P}xy \wedge \forall z \ ((\mathrm{P}zy \wedge z \subseteq_t x) \rightarrow \mathrm{P}zx)$

In order to prevent the proliferation of arbitrary entities, we only impose the existence of a slice of an entity when it temporally intersects another (so that we state only the existence of *meaningful* temporal parts, corresponding to temporal interaction between other entities). Thus we want:

$$\forall w, y(w\sigma y \rightarrow \exists u(TSuw \ \wedge \ u \subseteq_t y)) \qquad (1)$$

Since $(w\sigma y \rightarrow \exists u(u \subseteq_t w \ \wedge \ u \subseteq_t y))$, it is enough to state the following axiom:

A 4.6 $y \subseteq_t w \rightarrow \exists u \ (\mathrm{TS}uw \ \wedge \ u \equiv_t y)$

and (1) then is a theorem, along with the following ones:

Th 4.1 $\mathrm{P}xy \rightarrow \exists z(\mathrm{TS}zy \ \wedge \ z \equiv_t x)$
(there is a temporally equivalent slice for all parts of an entity)

Th 4.2 $(\mathrm{TS}xy \wedge \mathrm{TS}zy \ \wedge \ x \equiv_t z) \rightarrow x = z$
(this t-equivalent slice is unique)
We will note $w_{/y}$ the part of w corresponding to the "lifetime" of y when it exists (that is when $y \subseteq_t w$).

Models for the base theory of Asher & Vieu are presented in [Asher and Vieu, 1995]. We present here the properties that a sub-part of $\mathcal{P}(\mathbb{R}^2)$ needs to be a model of our theory; lack of space precludes the presentation of the proof of the consistency of each axiom with respect to this model, which in most cases is rather straightforward. The construction can be easily generalized to show that equivalent parts of $\mathcal{P}(\mathbb{R}^3)$ or $\mathcal{P}(\mathbb{R}^4)$ are models of the theory. At this stage we do not show the kinds of model with respect to which there is semantic completeness, we merely introduce a model that shows the consistency of our theory.
Let's consider a subset $\mathcal{F} \subset \mathcal{P}(\mathbb{R}^2)$. Asher and Vieu have shown in [Asher and Vieu, 1995] the consistency and completeness of their theory with respect to a certain kind of structure. Models for our own theory will have some of these properties, which we present now.
Consider a topological space $\mathcal{E}$. Objects of the theory denote elements of a set $\mathcal{F} \subset \mathcal{P}(\mathcal{E})$. The interior and closure operators are noted "int" and "cl" (in our case $\mathcal{E}$ will be $\mathbb{R}^2$ with a classical topology). The operator $\cup^*$ and $\cap^*$ are union and intersection operators preserving the "regularity" of interiors and closures:
$x \cup^* y = x \cup y \cup int(cl(x \cup y))$
$x \cap^* y = x \cap y \cap cl(int(x \cap y))$
In order to satisfy the axioms of the topological part of the theory, elements of $\mathcal{F}$ must verify the following conditions:

i. if $X \in \mathcal{F}$ and $Y \in \mathcal{F}$ and $int(X \cap Y) \neq \emptyset$, then $X \cap^* Y \in \mathcal{F}$. (The intersection $\cap^*$ of two elements of $\mathcal{F}$ is in $\mathcal{F}$ if it has a non empty interior).

ii. if $X \in \mathcal{F}$ and $Y \in \mathcal{F}$, $X \cup^* Y \in \mathcal{F}$. (The union of two elements of $\mathcal{F}$ is in $\mathcal{F}$)
iii. $\mathcal{E} \in \mathcal{F}$.
iv. if $X \in \mathcal{F}$ and $X \neq \mathcal{E}$ then $(C(X)/_{\mathcal{E}}) \in \mathcal{F}$ (if the complement of X is not empty, it is in $\mathcal{F}$)
v. Regularity: the interior of an element of $\mathcal{F}$ is not empty and "full" ($int(cl(X)) = int(X)$) and is in $\mathcal{F}$; its closure is regular ($cl(int(X)) = cl(X)$) and belongs to $\mathcal{F}$.

The interpretation of the relation of connection in this class of models for any variable assignation g, is as follows: $[\![Cxy]\!]_g$ =true iff $[\![x]\!]_g \cap [\![y]\!]_g \neq \emptyset$.

Now let's have a look at the added constraints on our models. Let $p_t(X)$ be the projection of $X \in \mathcal{F}$ on the temporal dimension:

$$p_t(X) = \{a \in \mathbb{R} / \exists b \in \mathbb{R},\ (b, a) \in X\}$$

We give the following interpretation to the primitives < and $\asymp$. Let g be an arbitrary variable assignment:
$[\![x \asymp y]\!]_g = 1$ iff there is $a \in \mathbb{R}$, $b \in \mathbb{R}$ et $c \in \mathbb{R}$ such that $(a, c) \in [\![x]\!]_g$ and $(b, c) \in [\![y]\!]_g$.
$[\![x < y]\!]_g = 1$ iff for all real numbers a,b,c,d, $((a, b) \in [\![x]\!]_g \wedge (c, d) \in [\![y]\!]_g) \rightarrow b < d$
We add the following constraints:

α) for all $X \in \mathcal{F}$ and $Y \in \mathcal{F}$ such that $p_t(X) \subset p_t(Y)$ we have also $Y \cap (\mathbb{R} \times p_t(X)) \in \mathcal{F}$
This constraint is easily compatible with the others, since $Y \cap (\mathbb{R} \times p_t(X))$ has a non-empty interior since Y and X (and therefore $\mathbb{R} \times p_t(X)$) have non-empty interiors.

β) there are $X \in \mathcal{F}$ and $Y \in \mathcal{F}$ such that for all $(a, b) \in X$ and $(c, d) \in Y$, $b < d$.

γ) there are $X \in \mathcal{F}$ and $Y \in \mathcal{F}$ such that $p_t(X) \cap p_t(Y) \neq \emptyset$ et $X \cap Y = \emptyset$

The constraint α, β, γ are necessary because of (respectively) axioms A4.6, A4.3 and A4.2.

4.2 CONTINUITY

We have already mentioned the importance of continuity in our intuitive understanding of motion (motion is a perceptually continuous spatial change), and how this notion pervades a lot of work on spatial relations, the most obvious being the transition graphs (of RCC8 [Randell *et al.*, 1992], e.g.), without being given a generic formal characterization. The continuity we have in mind (along with other authors in QSR) is different from the mathematical sense where arbitrarily fine distinctions can be made about space and time. We want something closer to intuition and at the same time covering the properties shown to be necessary for a theory of motion. The continuity we propose can be defined within the theory without stating separately the possible transitions, contrarily to what Galton does in [Galton, 1993][2], and this provides a more general characterization of qualitative continuous change; it is thus moreover much easier to check the consistency of such a theory. Informally, we propose to consider a spatio-temporal individual as qualitatively continuous if it is temporally self-connected and if it doesn't make spatial "leaps" (corresponding to a sudden gain or loss of parts, or a sudden translation), that is if no part of an entity can be temporally connected to a slice of that entity but not spatio-temporally (a counter-example of this is illustrated in Figure 5, where the non-continuity of w corresponds to a qualitative "horizontal" jump; this figure also illustrates theorem 4.3 below and the entity z can be ignored at this stage of the discussion). This can be expressed

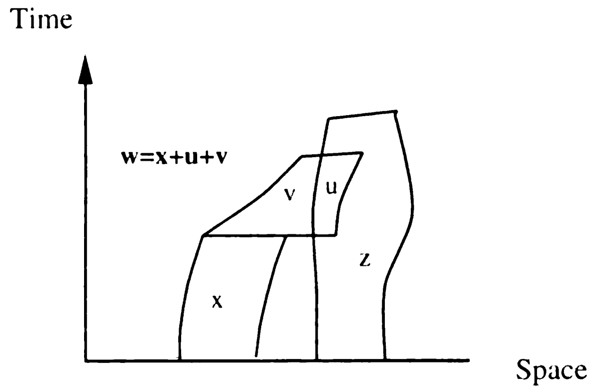


Figure 5: Entity w is not Continuous.

as follows:

D 4.2
CONTINU$w \triangleq$CON$_t w \wedge \forall x \forall u((\mathrm{TS}xw \wedge x \asymp u$
$\wedge\ \mathrm{P}uw) \rightarrow \mathrm{C}xu)$

We will present now what we recover from the intuitions about continuous change. We will compare spatio-temporal relations between two contemporaneous slices of two "continuous" individuals at two different moments, in order to compare the results with RCC conceptual neighborhood graph. Figure 6 shows an example of what this means: during z_1, the corresponding slices of x and y are disconnected, later on, during z_2 the corresponding slices of x and y "spatially" overlap, i.e. every temporal slices of x and y overlap during z_2, which could be expressed as
$\mathrm{O}_{sp}xy \triangleq \mathrm{O}xy \wedge x \subseteq_t (x \cdot y)$; this way, we can define all "spatial" relations corresponding to those definable in RCC (it is to be noted that DC is already equivalent to DC$_{sp}$). We

[2]Moreover we express the continuity of transition for any entity whereas Galton focused on rigid objects, a concept which cannot be expressed in a topological theory, and which eliminates some transitions such as NTPP to =.

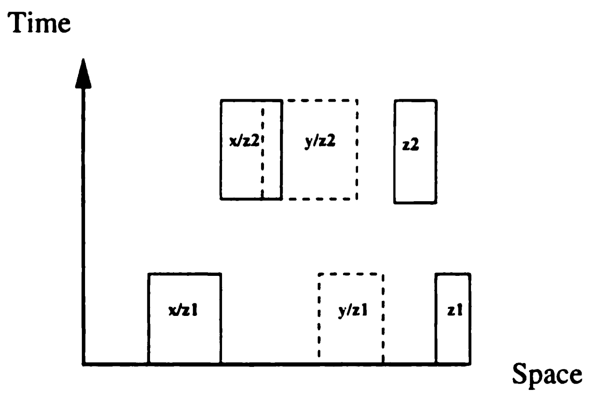


Figure 6: Purely "Spatial" Relations

found that impossible transitions correspond to the impossible transitions of the conceptual neighborhood graph of RCC8, with one exception: if the theory allows for atoms, continuity as defined here cannot apply to them since they have no parts [3]. Some transitions are moreover only possible between closed entities (a property which is assumed for RCC classical conceptual neighborhoods, since the theory cannot distinguish open and closed entities). In order to recover Figure 5, it suffices to demonstrate the following theorems (every time we mention a transition between two RCC relations they have to be understood as their purely spatial counterparts in our framework):

i) A continuous transition between DC and O is impossible (thus a continuous transition between DC and PO or TPP or NTPP or TPPI or NTPPI or "=" is impossible).

ii) A continuous transition between EC and NTP or TP is impossible (and thus NTPP and TPP and =).

iii) A continuous transition between PP and PP^{-1} is impossible (so that no transition from TPP or NTPP to TPPI or NTPPI is possible)

iv) A continuous transition between PO and NTPP is impossible. PO being symmetric, the impossibility of a $PO/NTPP^{-1}$ is proved at the same time.

This way, the only remaining possibilities are the transition of Figure 7.

The non-continuity of a DC↔O transition

Th 4.3 $(\mathrm{TS}xw \wedge \mathrm{P}yw \wedge \mathrm{DC}xz \wedge \mathrm{O}_{sp}yz \wedge x \asymp y) \rightarrow \neg\mathrm{CONTINU}w$

[3]The problem of atoms should be dealt with separately since it is not clear what properties they should have; few authors have addressed those questions, but we think it is rather natural for atoms to be left out in what is presented here.

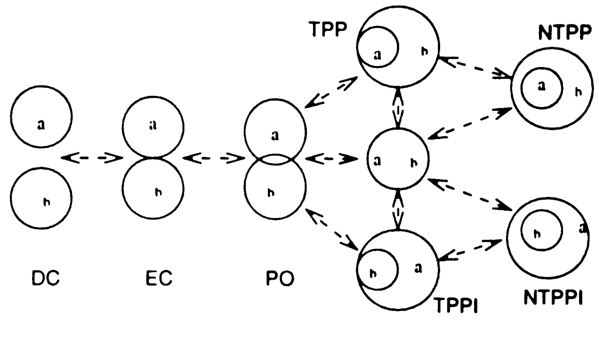


Figure 7: RCC8 Conceptual Neighborhoods

This says that a direct transition from DC (i.e. $\neg C$) and O cannot be continuous (see Figure 5). (Proof: easily, (with $u = y \cdot z$) $\neg\mathrm{C}xu$ et $x \asymp u$, and this proves w's non-continuity). The hypotheses of this theorem imply that x and y are closed (although in a non obvious way).

The non-continuity of a TP/NTP↔EC transition
This can be shown if we assume the entities involved to have only non-atomic temporal parts (that is if we assume x and y are GATs):

Th 4.4
$(\mathrm{TS}xw \wedge \mathrm{P}yw \wedge \mathrm{EC}xz \wedge \mathrm{TP}yz \wedge x \asymp y \wedge \neg GAT(x) \wedge \neg GAT(y)) \rightarrow \neg\mathrm{CONTINU}w$

More easily (and more generally), we also have:

Th 4.5 $(\mathrm{TS}xw \wedge \mathrm{P}yw \wedge \mathrm{EC}xz \wedge \mathrm{NTPP}_{sp}yz \wedge x \asymp y) \rightarrow \neg\mathrm{CONTINU}w$
(No continuous transition between EC and NTP).
See Figure 8 for the illustration of theorem 4.4. If x or y are generalized atoms, $x+y$ is still consistent with the definition of continuity. Note that the theorem mentions EC, which gives the intended result, given that $\mathrm{EC}_{sp} \rightarrow \mathrm{EC}$. The definition of $\mathrm{NTPP}_{sp}xy$ departs a little from the other "purely" spatial relation since we may want it to hold on temporally closed intervals, and thus it cannot entail $\mathrm{NTPP}xy$. We get around this by stating
$\mathrm{NTPP}_{sp}xy \triangleq \forall z(z \subseteq_t (ix) \rightarrow \mathrm{NTPP}x_{/z}y_{/z}$.

The non-continuity of a PP↔PP^{-1} transition
Observing that: $\mathrm{PP}xz \leftrightarrow \neg\mathrm{C}x(-z)$ and $\mathrm{PP}^{-1}yz \leftrightarrow (\mathrm{O}y(-z) \vee \mathrm{EC}y(-z))$ and using the previous theorems with $-z$ or $c(-z)$ in place of z, we show that there cannot be a direct continuous transition between PP and PP^{-1} (see Figure 9).

The non-continuity of a PO↔NTPP transition
We have (see Figure 10):
Th 4.6 $(z = z_1+z_2 \wedge \mathrm{TS}yw \wedge \mathrm{NTPP}_{sp}yz_1 \wedge \mathrm{P}xw \wedge \mathrm{PO}_{sp}xz_2 \wedge x \asymp y) \rightarrow \neg\mathrm{CONTINU}(w)$

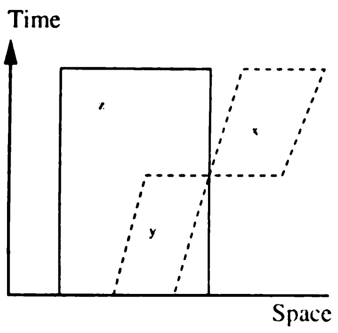


Figure 8: PP/EC Transition

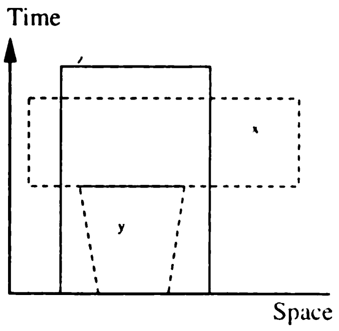


Figure 9: PP/PP^{-1} Transition

Indeed, considering $u = x - z$, we have $u \bowtie y$ and $\neg \mathrm{C}uy$.

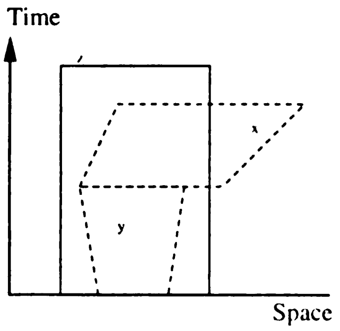


Figure 10: NTPP/PO Transition

To sum up this section, we have given a formal characterization of the conceptual neighborhood graph that is usually taken as an intuitive reality for RCC-like theories. We go further than Galton who had only listed extensively (in a different spatio-temporal setting) what this transitions should be in a limited case. We did this by defining spatial relations corresponding to the standard intended interpretations, even though theories differ as to what their exact models are. We have also shown the necessity of the distinction between open and closed regions and why we have chosen Asher and Vieu's topological framework.

5 MOTION CLASSES

5.1 REPRESENTING MOTION CLASSES

This section is dedicated to showing how the theory can be used to represent certain types of motion. This has been inspired from a study of some motion verbs [Muller and Sarda, 1997], and puts the expressive power of the theory to the test. We insist on the fact that we have restricted ourselves to the topological aspects of such motions and that the representations are consequently to be understood as a partial definition only of the semantics of those verbs, and as a guide for an intuitive (but formal) definition of meaningful topological motion classes.
In order to do so, we need to introduce a few preliminary notions. First, we define another spatio-temporal partial overlap as an overlap which is not spatial only (a slice of x has to be completely out of y at some time):

D 5.1 $\mathrm{STPO}xy \triangleq \mathrm{O}xy \wedge \neg(x \equiv_t x \cdot y) \wedge \neg(y \equiv_t y \cdot x)$

Th 5.1 $\mathrm{STPO}xy \rightarrow \exists u(\mathrm{TS}ux \wedge \neg \mathrm{C}uy)$

A few other definitions are needed:

D 5.2 $\mathrm{LOC}xy \triangleq \exists z(\mathrm{TS}zx \wedge \mathrm{P}zy)$
(a slice of x is part of y)

D 5.3 $\mathrm{TEMP\_IN}xy \triangleq \mathrm{LOC}xy \wedge \mathrm{STPO}xy$
(x is "temporarily" a part of y)

We define relations equivalent to Allen's relations on intervals [Allen and Hayes, 1985] (note that =,<, already exists as $\equiv_t$, <):

D 5.4 $\mathrm{MEET}xy \triangleq ix < iy \wedge x \bowtie y$

D 5.5
$\mathrm{START}xy \triangleq x \subseteq_t y \wedge \forall z(z < x \rightarrow z < y) \wedge \neg y \subseteq_t x$

D 5.6
$\mathrm{FINISH}xy \triangleq x \subseteq_t y \wedge \forall z(x < z \rightarrow y < z) \wedge \neg y \subseteq_t x$

D 5.7 $\mathrm{O}_t xy \triangleq x \sigma y \wedge \exists u(\mathrm{START}uy \wedge \mathrm{FINISH}ux)$

D 5.8
$\mathrm{DURING}xy \triangleq x \subseteq_t y \wedge \neg \mathrm{FINISH}xy \wedge \neg \mathrm{START}xy$

The converse relations can be defined in an obvious way, and will be noted MEET_i, START_i, FINISH_i, O_{ti}, DURING_i. We can now very simply express different motion classes, such as those mentioned in the beginning of the section. We assume for all those motions that the entities involved (intuitively, the arguments of a motion event) are continuous, without writing it each time in the definition.

The first argument corresponds to an entity z that "temporalize" the relation: the relation holds between $x_{/z}$ and $y_{/z}$, x and y being two other spatio-temporal regions. At a later stage, we could build a theory of physical objects where x and y would be the trajectories of two objects and z would be another entity giving the temporal extent of the relation (e.g. the spatio-temporal extension of an event). None of these interpretations are necessary to understand the following given relations, which are defined independently of any assumption about entities except that they are spatio-temporally extended. We define six classes of motion, named LEAVE, REACH, HIT, CROSS, INTERNAL and EXTERNAL; intuitively, each of these classes could be instantiated by the following motion verbs: LEAVE, REACH, HIT and CROSS correspond to the topological behavior of the corresponding verbs, INTERNAL corresponds to verbs such as "to drive around (the country)" and EXTERNAL to verbs such as "to avoid" (see Figure 11).

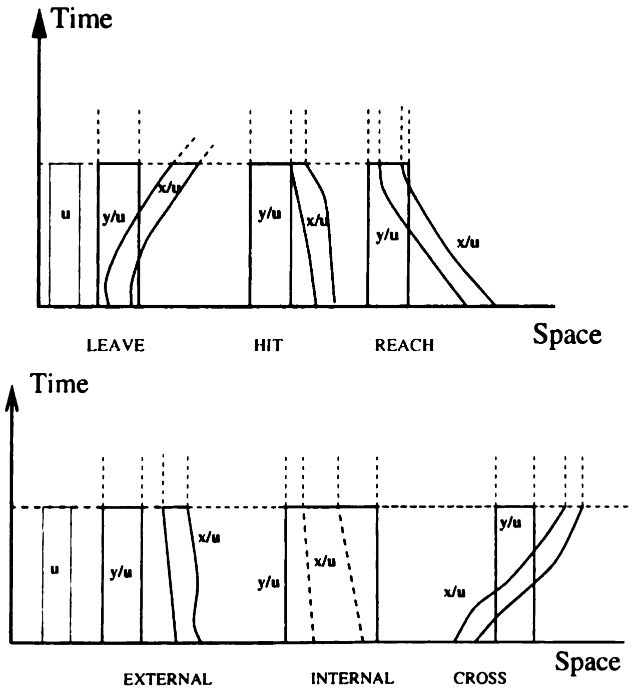


Figure 11: The Six Motion Classes

D 5.9
REACH$zxy \triangleq$TEMP_IN$x_{/z} y_{/z} \wedge$ FINISH$(x_{/z} \cdot y_{/z})z$

D 5.10
LEAVE$zxy \triangleq$TEMP_IN$x_{/z} y_{/z} \wedge$ START$(x_{/z} \cdot y_{/z})z$

D 5.11 INTERNAL$zxy \triangleq$PP$x_{/z} y_{/z}$

D 5.12
HIT$zxy \triangleq$EC$x_{/z} y_{/z} \wedge \forall x_1, y_1[(Px_1 x_{/z} \wedge$P$y_1 y_{/z} \wedge$EC$x_1 y_1)$
$\to$ (FINISH$x_1 z \wedge$FINISH$y_1 z$)]

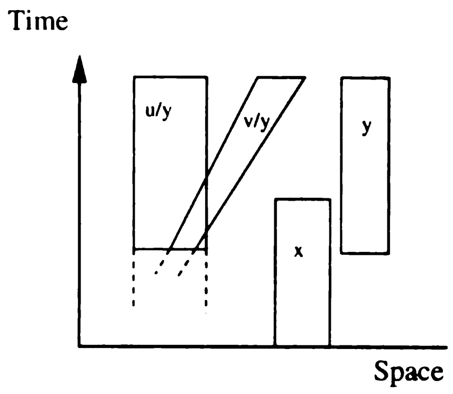


Figure 12: An Example for Time and Space-Time Composition

D 5.13 EXTERNAL$zxy \triangleq \neg$C$x_{/z} y_{/z}$

D 5.14
CROSS$zxy \triangleq \exists z_1, z_2(z = z_1+z_2 \wedge$ MEET$z_1 z_2 \wedge$
REACH$z_1 xy \wedge$ LEAVE$z_2 xy$)

5.2 REASONING ABOUT MOTION CLASSES

5.2.1 Combining Motion and Temporal Information

To be able to reason efficiently about spatial qualitative models, it is customary to use and implement composition tables, which are dedicated to more specific deductions than generic first-order theorem provers. To reason about motion, different paths can be followed. We show in Table 1-2 a composition of temporal and spatio-temporal information between two entities, following the pattern: ((R$_1 xy$ $\wedge$ Motion(y, u, v)) $\to$R$_2 u_{/x} v_{/x}$), R$_1$ and Motion being the entries in the table and R$_2$ being the corresponding result. This table takes as input information about the class of motion (Motion) and a temporal relation R$_1$ between the entity that determine the temporal extent during which the relation holds and any other entity (we choose to express this second piece of information with an Allen relation). The result is a spatio-temporal relation R$_2$ between the arguments (u and v) "during" the other entity (x)[4]. Figure 12 illustrates the kind of situation corresponding to the table, where R$_1$ is O$_t$ and Motion is LEAVE.

All entries of this table are theorems of the theory; entries assuming u and v are continuous are marked with a (c); we also assume entities are all GATs (see section 2). Space doesn't allow us to present the proofs here.

[4] We consider the cases of relations similar to RCC8, for the relation holding between $u_{/x}$ and $v_{/x}$, the resulting relation is a disjunction of one of the previously defined relations.

R_1/Motion	LEAVE	HIT	REACH
MEET (c)		DC,EC,PO	
O_t	PO,TPP, NTPP		DC,EC,PO
START		DC	
DURING	All		All
$\equiv_t$	PO	EC	PO
FINISH			
O_{ti}	PO,EC,DC		PO,TPP, NTPP
$MEET_i$ (c)		EC, PO	
$FINISH_i$			
$START_i$	PO		PO
$DURING_i$			

Table 1: Combining Motion and Temporal Information (1).

R_1/Motion	CROSS	INTERN.	EXTERN.
MEET (c)		PO, TPP, NTPP	DC, EC, PO
O_t	DC, EC, PO		
START			
DURING	All	PP	DC
$\equiv_t$	PO		
FINISH			
O_{ti}	DC, EC, PO		
$MEET_i$ (c)		PO, TPP, NTPP	DC, EC, PO
$FINISH_i$			
$START_i$	PO		
$DURING_i$			

Table 2: Combining Motion and Temporal Information (2).

5.2.2 Combining Motion and "Static" Information

Obviously, even though all objects are considered as having histories, we perceive our environment with respect to objects we regard as "static" and defining some kind of frame of reference. Thus motion is often considered as the motion of an entity relative to one or more given (often implicitly), "static" entities in a frame of reference. Thus we present now the combination of information about objects of a known environment (intuitively the spatio-temporal referent of locations in a frame of reference) and motion information. We will assume that some regions (corresponding to locations) bear the same relation to each other "during" any other entity. This means, that the relation R_1 in the Tables 3-4 is to be read as $R_1 y/_u z/_u \;\wedge\; \forall v(v \subseteq_t u \rightarrow R_1 y/_v z/_v)$ (this is what was meant by "purely" spatial relations, namely that the same relation holds between all respective temporal slices of those entities, cf section 4). Thus, from $R_1 y/_u z/_u$ and Motion(u, x, y), we can deduce a relation R_2 such that $R_2 x/_u z/_u$. Figure 13 shows an example entry of the table, where Motion is LEAVE and R_1 is DC. The result can be DC, EC or PO. This is obtained simply by combining the entries of Tables 1-2 with a composition table for topological relations. As the temporal information makes use of the continuity hypothesis, this is also only valid assuming that hypothesis. We want to point out at this stage, that we use those relations only as a means of comparison and that they do not really express as such all the information we want to convey when dealing with motion. It can be seen from the tables that (PO, TPP, NTPP) is a common result for very different motion relations (such as LEAVE and CROSS); we mainly wanted to focus here on why this theory is a good starting point for a theory of motion.

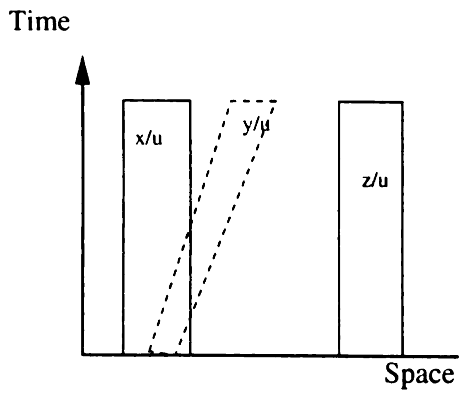


Figure 13: An Example of Location and Spatio-Temporal Combination

R_1/Motion	LEAVE	HIT	REACH
$DC_{\cdot p}$	DC,EC,PO	DC,EC,PO, PP^{-1}	DC,EC,PO
$EC_{\cdot p}$	DC,EC,PO	DC,EC,PO, TPP, $NTPP^{-1}$, TPP^{-1}	DC,EC, PO
$PO_{\cdot p}$	DC,EC,PO, TPP,NTPP	DC,EC,PO, TPP,NTPP	DC,EC,PO, TPP, NTPP
$TPP_{\cdot p}$	DC,EC,PO	DC,EC	DC,EC,PO
$NTPP_{\cdot p}$	DC,EC,PO	DC	DC,EC,PO
=	PO	EC	PO
$TPP^{-1}_{\cdot p}$	PO,TPP, NTPP	PO,TPP, NTPP	PO,TPP, NTPP
$NTPP^{-1}_{\cdot p}$	PO,TPP, NTPP	PO,TPP, NTPP	PO,TPP, NTPP

Table 3: Composing Motion and Location Information (2).

6 CONCLUSION

We have presented here an original relational theory combining space and time to achieve a theory of spatio-temporal entities and thus a qualitative theory of motion. It is possible to define a notion of qualitative spatio-temporal continuity within the theory which gives a rigorous basis to intuitions that seemed reasonable to admit about most RCC-

R_1/Motion	CROSS	INTERN.	EXTERN.
DC_{sp}	DC,EC,PO	DC	All
EC_{sp}	DC,EC,PO	DC,EC	All
PO_{sp}	All	All	All
TPP_{sp}	DC,EC,PO	All	DC
$NTPP_{sp}$	DC,EC,PO	All	DC
=	PO	TPP,NTPP	DC
TPP_{sp}^{-1}	PO,TPP, NTPP	NTPP	All
$NTPP_{sp}^{-1}$	PO,TPP, NTPP	NTPP	All

Table 4: Composing Motion and Location Information (2).

like theories of space, namely the admissible transitions between qualitative spatial relations, in a more unified and general framework than [Galton, 1993]. The expressive power of the theory allows for the definition of complex motion classes such as those expressed by motion verbs in natural language. This complexity is in itself a problem if one is to reason about such representations, so we have given a few examples of the kind of reasoning that can be of interest about motion relations and that can be demonstrated within the theory. It is to be noted however that "motion" ranges over a wide set of phenomena and it should be expected that varied courses of action can be followed in the task of automating reasoning about it, depending on the kind of task one is interested in (it can be qualitative reasoning about physical objects, such as [Hayes, 1985a, Davis, 1989] or representing natural language expressions; another area where it can be used is high-level vision systems dealing with natural language descriptions of spatial changes [Fernyhough *et al.*, 1998, Borillo and Pensec, 1995]). Besides we can now try to extend our theory to geometrical concepts (such as orientation and distance). Some work is now also needed to characterize completely the models of the theory (and then prove completeness with respect to those models).

Acknowledgements

I wish to thank the two anonymous reviewers for their helpful comments; Mario Borillo, Laure Vieu for continuous support; Pierre Gambarotto, Jérôme Lang, Claudio Masolo and Nicolas Maudet for inspiring discussions about the work presented here.

References

[Allen and Hayes, 1985] J. Allen and P.J. Hayes. A commonsense theory of time. IJCAI, 1985.

[Asher and Sablayrolles, 1995] N. Asher and P. Sablayrolles. A typology and discourse semantics for motion verbs and spatial PPs in French. *Journal of Semantics*, 12(1):163–209, June 1995.

[Asher and Vieu, 1995] N. Asher and L. Vieu. Towards a geometry of common sense: a semantics and a complete axiomatisation of mereotopology. In *Proceedings of IJCAI95*, 1995.

[Aurnague and Vieu, 1993] M. Aurnague and L. Vieu. A three-level approach to the semantics of space. In Zelinski-Wibbelt, editor, *Semantics of Prepositions in Natural Language Processing*, number 3 in Natural Language Processing, pages 393–439. Berlin : Mouton de Gruyter, 1993.

[Borillo and Pensec, 1995] M. Borillo and H. Pensec. From numerical observations to propositional representation: a cognitive methodology to structure hybrid spatial knowledge. Sheffield, 1995. AISB proceedings.

[Carnap, 1958] Rudolf Carnap. *Introduction to Symbolic Logic and its Applications*. Dover, New York, 1958.

[Clarke, 1981] B. Clarke. A calculus of individuals based on 'connection'. *Notre Dame Journal of Formal Logic*, 22(3):204–218, July 1981.

[Davis, 1989] E. Davis. *Representations of Commonsense Knowledge*. Morgan Kaufmann, 1989.

[Fernyhough *et al.*, 1998] J. Fernyhough, A.G. Cohn, and D.C. Hogg. Building qualitative event models automatically from visual input. In *Proc. Int. Conf. on Computer Vision (ICCV98)*, pages 350–355. IEEE, Narosa Publishing House, 1998.

[Fleck, 1996] M.M. Fleck. The topology of boundaries. *Artificial Intelligence*, 80:1–27, 1996.

[Forbus, 1983] K. Forbus. Qualitative reasoning about space and motion. In D. Gentner and A.L. Stevens, editors, *Mental models*, pages 53–73. Erlbaum, Hillsdale (NJ), 1983.

[Galton, 1993] A. Galton. Towards an integrated logics of space, time and motion. IJCAI, 1993.

[Galton, 1995] A. Galton. Time and change for IA. In Gabbay, editor, *Logics for Epistemic and Temporal Reasoning*, volume 4 of *Handbook of Logics for AI and Logic Programming*. Oxford University Press, 1995.

[Galton, 1997] A. Galton. Space, time and movement. In O. Stock, editor, *Spatial and Temporal Reasoning*. Kluwer, 1997.

[Hartley, 1992] R. Hartley. A uniform representation for time and space and their mutual constraints. *Computers Math. Applic.*, 1992.

[Hayes, 1985a] P.J. Hayes. An ontology for liquids. In J.R. Hobbs et R.C. Moore, editor, *Formal Theories of the Commonsense World*. Ablex Publishing Corporation, Norwood, 1985.

[Hayes, 1985b] P.J. Hayes. The second naive physics manifesto. In J.R. Hobbs et R.C. Moore, editor, *Formal Theories of the Commonsense World*, pages 1–36. Ablex Publishing Corporation, Norwood, 1985.

[Hays, 1989] E. Hays. On defining motion verbs and spatial prepositions. Technical report, Universitat des Saarlandes, 1989.

[Kamp, 1979] H. Kamp. Events, instants and temporal reference. In Von Stechow Bäuerle, Egli, editor, *Meaning, use and interpretation of language*, pages 376–417. de Gruyter, Berlin, 1979.

[Muller and Sarda, 1997] P. Muller and L. Sarda. The semantics of french transitive movement verbs and the ontological nature of their objects. In *Proceedings of ICCS'97*, Donostia-San Sebastian, May 1997.

[Rajagopalan and Kuipers, 1994] R. Rajagopalan and B. Kuipers. Qualitative spatial reasoning about objects in motion: application to physics problem solving. San Antonio, TX, March 1994. IEEE Conference on Artificial Intelligence for Applications (CAIA-94).

[Randell *et al.*, 1992] D. Randell, Z. Cui, and A. Cohn. A spatial logic based on regions and connection. San Mateo CA, 1992. KR'92, Morgan Kaufmann.

[Russell, 1914] Bertrand Russell. *Our Knowledge of the External World.* Routledge, London and New York, 1914.

[Shanahan, 1995] M. Shanahan. Default reasoning about spatial occupancy. *Artificial Intelligence*, 74:147–163, 1995.

[Simons, 1987] Peter Simons. *Parts - A Study in Ontology*. Oxford University Press, Oxford, 1987.

[van Benthem, 1995] J. van Benthem. Temporal logic. In Gabbay, editor, *Logics for Epistemic and Temporal Reasoning*, volume 4 of *Handbook of Logics for AI and Logic Programming*. Oxford University Press, 1995.

[Varzi, 1996] A. Varzi. Parts, wholes, and part-whole relations: The prospects of mereotopology. *Data and Knowledge Engineering*, 20(3):259–286, 1996.

[Vieu, 1991] L. Vieu. *Sémantique des relations spatiales et inférences spatio-temporelles: une contribution à l'étude des structures formelles de l'espace en langage naturel.* PhD thesis, Université Paul Sabatier, Toulouse, 1991.

[Vieu, 1997] L. Vieu. Spatial representation and reasoning in AI. In O. Stock, editor, *Spatial and Temporal Reasoning*, pages 3–40. Kluwer, 1997.

Diagnosis

On the Compilability of Diagnosis, Planning, Reasoning about Actions, Belief Revision, etc.

Paolo Liberatore
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma - Italy
Email: liberato@dis.uniroma1.it
WWW: http://www.dis.uniroma1.it/~liberato/

Abstract

In this paper we investigate the usefulness of preprocessing part of the input of a given problem to improve the efficiency. We extend the results of [Cadoli *et al.*, 1996] by giving sufficient conditions to prove the unfeasibility of reducing the on-line complexity via an off-line preprocessing. We analyze the problems of diagnosis [Peng and Reggia, 1986], planning [Bylander, 1991], reasoning about actions [Gelfond and Lifschitz, 1993], and belief revision [Williams, 1994]. We analyze other problems from various fields.

1 INTRODUCTION

Many problems in Artificial Intelligence (and in Knowledge Representation in particular) are computationally hard to solve. An observation that may lead to a reduction of the complexity is that often these problems have a property: the input is divided in two parts, where one of them is known long before the rest of the input. In such cases, it makes sense to do some computation on the first part of the input to speed-up the solution of the problem when the second part arrives. This computation made in advance is said *preprocessing*, or *compilation*. Another case in which a preprocessing is useful is when there are many instances of the problem to be solved that share part of the input (i.e. there are many instances for which part of the input is common). In such cases, the preprocessing of the shared part is useful if it decreases the cost of solving each single instance. We call the part of the input we preprocess *fixed part*, while the rest of the input is called *varying part*. When the complexity of the problem decreases thanks to the compilation of the fixed part, we call the problem *compilable*.

The general idea of preprocessing part of the input data has been widely used in many areas of computer science. For example, in computational geometry the techniques for geometric searching (cf. [Preparata and Shamos, 1985, Chap. 2]) use a costly preprocessing phase to speed up the on-line behavior.

In AI, a clear example of a problem with a fixed/varying part pattern occurs in the diagnosis field. Suppose we have a set of possible faults, and a set of rules that, given a fault, determine which (observable) effects will occur. In diagnosis of physical systems, the set of faults is the set of possible malfunctioning parts, while the effects are the (observable) abnormal behaviors of the system. The problem is to determine a minimal explanation (a minimal set of faults) that explains a given set of effects. It is reasonable to assume that the physical system (and its description in terms of faults-effects rules) is known in advance, and thus a preprocessing of it is in many case affordable (even if it is very expensive), in order to obtain a more efficient algorithm to find a diagnosis when a set of effects occurs.

Recently, several papers [Cadoli *et al.*, 1995, Gogic *et al.*, 1995, Kautz and Selman, 1992] focused on techniques to prove that problems can (or cannot) be compiled. The precise formalization of compilability is given in [Cadoli *et al.*, 1996].

What's new. In this paper, we give new sufficient conditions to prove that problems are not compilable, and apply them to the problems of diagnosis [Peng and Reggia, 1986], planning [Bylander, 1991], and reasoning about actions [Gelfond and Lifschitz, 1993]. Some problems of belief revision [Williams, 1994] will be proved to be compilable.

We also show the generality of these sufficient conditions by applying them to problems from database theory [Chandra and Merlin, 1977], network analysis and design [Garey and Johnson, 1979], software engineering [Ntafos and Hakimi, 1979], and graph theory [Karp, 1972].

Outline. The paper is organized as follows: in the next section, we report the basic concepts of compilability, as defined in [Cadoli *et al.*, 1996]. In Section 3 we show some new sufficient conditions to prove that a problem is not compilable. The other sections contain the analysis of various problems from the point of view of compilability.

2 DEFINITIONS

In this paper we consider propositional formulas, (directed) graphs, and languages. We assume that the basic definitions of these concepts are known. Problems are represented by languages (as usual) over an alphabet Σ. The *length* of a string $x \in \Sigma^*$ is denoted $||x||$. The problem of diagnosis, which we will use to illustrate the rationale of compilation, is defined as follows.

Definition 1 *The problem Minimal Diagnosis* (md) *is: given a set of faults F, a set of effects M, a function $e : F \to \mathcal{P}(M)$, a set of observed effects $M' \subseteq M$, and an integer k, decide if there exists a set $F' \subseteq F$ of size at most k, such that $\bigcup_{f \in F'} e(f) \supseteq M'$.*

We illustrate this definition with an example that will be also useful for explaining the concepts of compilability.

Example 1 *A car may overheat for (at least) four causes: the water pump is broken, the sensor is broken, the relay does not work, or there is not enough water in the radiator. Each of these faults generates an overheating. They may have other effects: if the water pump is broken, the radiator is not heat, while a fault in the sensor or the relay causes the fan not to turn on. On the contrary, the only effect of not having enough water in the radiator is the overheating*[1].

The set of faults and manifestations are defined as follows.

$$\begin{aligned} F &= \{\texttt{pump}, \texttt{sensor}, \texttt{relay}, \texttt{water}\} \\ M &= \{\texttt{overheat}, \texttt{fan\_off}, \texttt{no\_heat\_radiator}\} \end{aligned}$$

The function e, expressing the effects of faults, is defined as follows.

$$\begin{aligned} e(\texttt{pump}) &= \{\texttt{overheat}, \texttt{no\_heat\_radiator}\} \\ e(\texttt{sensor}) &= \{\texttt{overheat}, \texttt{fan\_off}\} \\ e(\texttt{relay}) &= \{\texttt{overheat}, \texttt{fan\_off}\} \\ e(\texttt{water}) &= \{\texttt{overheat}\} \end{aligned}$$

If the observed effects are $M' = \{\texttt{overheat}, \texttt{fan\_off}\}$, then the minimal diagnoses are $\{\texttt{sensor}\}$ and $\{\texttt{relay}\}$. Thus, there exist diagnoses of size $k = 1$. On the contrary, if the observed effects are $M' = \{\texttt{overheat}, \texttt{fan\_off}, \texttt{no\_heat\_radiator}\}$, then the minimal diagnoses are $\{\texttt{pump}, \texttt{sensor}\}$ and $\{\texttt{pump}, \texttt{relay}\}$, thus there is no diagnosis of size less or equal than $k = 1$.

In this paper we are concerned with problems whose input can be divided in two parts. One part is accessible off-line, that is, we can preprocess it. This part is called fixed part. The rest of the input is called varying part. In the example of diagnosis, we can spend much time of computation on the description of the physical system, while once a set of effects occurs, we want to obtain a diagnosis very quickly (in real time, if possible).

Formally, problems whose input is composed of two parts can be represented by sets of pairs of strings $S = \{\langle x, y\rangle\}$, where the first element of each pair is the fixed part of the input. In the case of diagnosis, the first part of the input is the description of the system, while the varying part is the set of observed effects. Since the function e represent the description of the system, while M' and k depend on the observed effect, we express Minimal Diagnosis as a problem of pairs of strings as follows.

$$\begin{aligned} \mathsf{md} \;=\; & \{\langle e, (M', k)\rangle \mid \exists F' \text{ of size at most } k, \\ & \text{such that } \bigcup_{f \in F'} e(f) \supseteq M'\} \end{aligned}$$

We do not write F and M for the sake of simplicity, and assume that F and M are simply the domain and the co-domain of the function e.

We remark that "fixed" is different from a more common concept of computational complexity, "constant". In computational complexity, the word "constant", when refers to a string, means that the string will always be of small length. On the contrary, "fixed" means that the part of the input will be known to the preprocessor, while the varying part is not. A fixed part, in our definition, may be arbitrarily large. This holds both in cases in which the fixed part is known in advance and in cases in which many input instances share the same fixed part.

In order to formalize the concept of preprocessing, we recall the definitions of polysize functions, compilable

[1]Some of the facts reported here are true, but since we are not really expert about cars, the reader should not try to repair cars following such diagnostic rules.

classes and reductions. These concepts have been introduced in [Cadoli *et al.*, 1996], where more examples and motivations are provided. A function f is called *polysize* if there exists a polynomial p such that for all x it holds $||f(x)|| \leq p(||x||)$, that is, the *size* of the result of f is bounded by a polynomial in the size of x.

Our intuitive notion of compilability states that a problem S can be reduced into a simpler problem by preprocessing only its fixed part. Given a complexity class C, we introduce the class of problems compilable into C, denoted by compC. The class we mostly use is compP, which contains all the problems that become polynomial, once the preprocessing is done.

Definition 2 *A language of pairs of strings*[2] $S \subseteq \Sigma^* \times \Sigma^*$ *belongs to* compC *if and only if there exist a polysize function* f *and a language of pairs* S' *such that for all* $x, y \in \Sigma^*$ *we have that*

1. $\langle x,y \rangle \in S$ *iff* $\langle f(x), y\rangle \in S'$.
2. $S' \in C$.

Notice that no restriction is imposed on the *time* needed to compute f, but only on the size of the result. This definition can be represented in terms of a computing machine as in Figure 1.

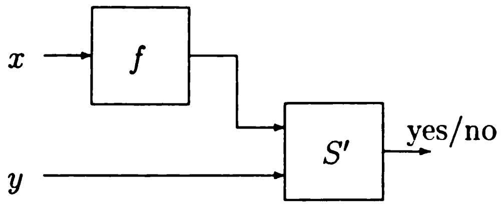


Figure 1: The compC machine.

This machinery captures our intuitive notion of compilability into C of a problem S with fixed and varying parts. We process off-line the fixed part x, obtaining $f(x)$. Now we can decide $\langle f(x), y\rangle \in S'$ with an algorithm in C. The whole process is convenient if the latter is easier than deciding $\langle x,y\rangle \in S$. In this paper we use mainly the class compP, that is the class of problems that can be solved in polynomial time after a preprocessing of the fixed part. The condition that f must be polysize is due to the fact that the fixed part x of the input may be arbitrarily large. If x were small, there would be no problem in having $f(x)$ of exponential size. Since x may be large, the condition ensures that the result of the preprocessing $f(x)$ will be of reasonable size.

[2]In the sequel we omit the two words "of strings": in the sequel, languages of pairs are always languages of pairs of strings.

Let us show how the idea of compilation can be used in the scenario of Example 1.

Example 2 *Consider the diagnosis problem of Example 1. The problem can be solved by considering each possible set* $F' \subseteq F$ *with size at most* k, *and verifying whether the effects of* F' *includes* M'. *In our example, we should check each subset of* F = {`pump`, `sensor`, `relay`, `water`}.

The complexity of the problem can be reduced via a compilation of the function e. Since e is known much in advance, we can preprocess it. The result of the compilation is a table, whose rows are pairs (set of manifestations, minimal size of diagnosis). When applied to the problem of the car, the result of the compilation is the following.

{`overheat`}	1
{`fan_off`}	1
{`no_heat_radiator`}	1
{`overheat`, `fan_off`}	1
{`overheat`, `no_heat_radiator`}	1
{`fan_off`, `no_heat_radiator`}	2
{`overheat`, `fan_off`, `no_heat_radiator`}	2

The table can be built by considering each possible $F' \subseteq F$ *and then calculating its effects. This is very inefficient, because we need to consider each possible* F'. *However, this is done during the compilation phase, and thus we are not very concerned about efficiency. When a set of manifestation occurs, the problem can be solved by a single check in the table. If the set of manifestations is for example* {`overheat`, `fan_off`}, *the size of the minimal diagnosis can be determined by looking up the table for a row whose first element is the set* {`overheat`, `fan_off`}. *Since the second element of the row is* 1, *there is a diagnosis of size* 1.

Note that the table has been calculated during the compilation phase, only knowing the function e. The solution of a specific instance of the problem can then be determined very efficiently. In this case, we were able to build the table because M was small. In general, such a table requires 2^m *rows, where m is the number of possible manifestations. While an exponential running time may be considered affordable during the compilation phase, it is always impossible to store an exponential-size table.*

So far, we have defined when a problem is compilable. What is missing is a way to prove that a problem *is not* compilable. Let us consider how this is done for general problems (without a fixed/varying part pattern). The usual way to prove intractability (that is, the non-membership in the class P) of a problem is to give a polynomial reduction from an NP-hard problem to it. Since language of pairs of strings can also been seen as problems of strings, this definition can be specialized for languages of pairs.

Definition 3 *A polynomial reduction from a language of strings R to a language of pairs (of strings) S is a pair of polynomial functions $\langle r, h\rangle$ such that, for any $x \in \Sigma^*$ it holds $x \in R$ iff $\langle r(x), h(x)\rangle \in S$.*

This is the obvious specialization of the usual definition of polynomial reduction to the case in which the target language is a language of pairs. The following is the specialization of the definition of hardness. We use these definitions in the next section.

Definition 4 *A language of pairs S is C hard if any language R in C can be polynomially reduced to it.*

The NP hardness of a problem proves that the problem is not polynomial. However, the NP hardness of a problem of pairs S does not suffice to show its incompilability: an NP hard problem may be very well compilable into P. On the converse, the main aim of compilation is to solve NP hard problems in polynomial time, using a preprocessing of part of the input.

In order to obtain a class of reductions that preserve the compilability property and are powerful enough to allow the definition of complete problems, we introduce the notion of *comp reduction*:

Definition 5 *A* comp reduction *between two problems of pairs A and B is a triple $\langle f_1, f_2, g\rangle$ such that f_1 and f_2 are polysize functions, g is a polynomial function, and for every pair of strings $\langle x, y\rangle$ it holds*

$$\langle x, y\rangle \in A \quad \textit{iff} \quad \langle f_1(x), g(f_2(x), y)\rangle \in B$$

If there exists a comp reduction between two problems of pairs A and B we say that A is *comp reducible* to B.

Intuitively, A is comp reducible to B if 1) the fixed part of B can be obtained from the fixed part of A using a polysize function (f_1), and 2) the variable part of B can be constructed using both a polysize function (f_2) applied to the fixed part of A, and a polynomial-time function (g) applied to the variable part of A. These reductions satisfy all the basic properties of reductions. Indeed, if C is a class of the polynomial hierarchy (e.g. P, NP), we have that:

Theorem 1 *The comp reductions are transitive and compatible*[3] *with the class* compC.

Therefore, it is possible to define a notion of *completeness* for compC.

Definition 6 *Let B be a language of pairs. B is* compC*-hard iff all problems $A \in$* compC *are comp reducible to B.*

Using the above definition we can find hard problems. Nevertheless, for many natural problems the non-compilability proofs appeared in the literature [Kautz and Selman, 1992, Cadoli *et al.*, 1997, Cadoli *et al.*, 1995, Gogic *et al.*, 1995] cannot be rephrased as proofs of compNP hardness. For example, the problem of diagnosis introduced in the last section is not compNP hard. However, we can prove its incompilability (its non-membership in compP), using the non-uniform classes and reductions. For languages of strings, the non-uniform classes were introduced by Karp and Lipton [1980]. The extension for languages of pairs is given in [Cadoli *et al.*, 1996], where the non-uniform compilability classes are also compared with Karp and Lipton's classes.

Definition 7 *A language of pairs S belongs to* nucompC *(non-uniform* compC*) iff there exists a polysize function f and a language of pairs S' such that for all $\langle x, y\rangle$ we have that:*

1. $\langle x, y\rangle \in S$ *iff* $\langle f(x, ||y||), y\rangle \in S'$;
2. $S' \in C$.

In order to prove the non-compilability of problems, the concepts of nucomp reduction and hardness are defined.

Definition 8 *A nucomp reduction (non-uniform comp reduction) between two problems of pairs A and B is a triple $\langle f_1, f_2, g\rangle$ such that f_1 and f_2 are polysize functions, g is a polynomial function, and for each pair $\langle x, y\rangle$ it holds*

$$\langle x, y\rangle \in A \quad \textit{iff} \quad \langle f_1(x, ||y||), g(f_2(x, ||y||), y)\rangle \in B$$

We say that A is *nucomp reducible* to B if there exists a nucomp reduction from A to B. The definition of hardness follows.

Definition 9 *A problem of pairs B is said* nucompC *hard (non-uniform* compC *hard) if any problem in* nucompC *is nucomp reducible to it B.*

[3]For the definition of compatibility see [Johnson, 1990, pg. 79].

Now, from [Cadoli *et al.*, 1996] it follows that a nucompNP hard problem cannot be compiled into P, unless $\Pi_2^p = \Sigma_2^p$ (that is, unless the polynomial hierarchy collapses).

The concepts defined in this section might look complex. However, what is really needed for understanding the sequel of the paper can be summarized in the following small box.

S is in compP $\Rightarrow$ S can be compiled into P

S is nucompNP hard $\Rightarrow$ S cannot be compiled into P

A problem S can be compiled into P if it can be solved with a polynomial algorithm, after preprocessing part of the input.

The nucompNP hardness of a problem is the way to prove that a problem cannot be compiled into P. However, this requires to prove that there is a nucomp reduction from a previously proved nucompNP hard problem. This is not simple. The proofs of nucompNP hardness (the nucomp reductions) are usually complex and hard to find. In the next section we give some sufficient conditions to prove the nucompNP hardness of problems.

3 MONOTONIC POLYNOMIAL REDUCTIONS AND DIAGNOSIS

In this section we give new sufficient conditions for proving the nucompNP hardness of problems. These conditions are based on the concept of monotonic polynomial reductions, which are particular polynomial reductions from languages of strings to languages of pairs. The use of monotonic reductions greatly simplifies the proofs of incompilability. The proofs of nucompNP hardness that can be found in the literature [Cadoli *et al.*, 1995, Cadoli *et al.*, 1997] are often long and complex. We show in the sequel how the proofs based on monotonic reductions are instead very simple and intuitive.

A proof of NP hardness of a language S is usually a polynomial reduction from a (previously) proved NP hard problem R to S. Suppose, for example, that we have proven the NP hardness of a language of pairs S by means of a polynomial reduction from the problem 3sat. We state the following definition.

Definition 10 *A polynomial reduction $\langle r, h\rangle$ from* 3sat *to a language of pairs S is said to be monotonic if, for any two sets of clauses Π_1 and Π_2 over the same alphabet, with $\Pi_1 \subseteq \Pi_2$, it holds*

$$\langle r(\Pi_1), h(\Pi_1)\rangle \in S \quad \Leftrightarrow \quad \langle r(\Pi_2), h(\Pi_1)\rangle \in S \qquad (1)$$

It can be proved that, given a monotonic polynomial reduction, one can build a proof of nucompNP hardness for the problem S.

Theorem 2 *If there exists a monotonic polynomial reduction from* 3sat *to a problem of pairs S, then S is* nucompNP *hard.*

As a result, in order to prove that a problem S is not compilable, it suffices to find a polynomial reduction from 3sat to S, and then to prove that this reduction has the property of monotonicity (1). This is useful, because often there already exists a proof of hardness of the problem S that uses a reduction that can be easily proved to be monotonic.

The following theorem, and its proof, shows how this result can be applied to the problem of diagnosis. The inputs of the problem are the set of all the possible faults and effects F and M, the function e, the set of the observed effects M' and the number k. The set of possible effects, faults, and the function e are the fixed part (since it is determined by a static analysis of the system to be diagnosed), while the set of the current effect (that is, the abnormal behavior to be analyzed) and k are the varying part.

Theorem 3 *The problem* md *is* nucompNP *hard (and thus it is not in* compP*).*

Proof. The problem md has been proved to be NP hard via the following reduction from 3sat [Allemang *et al.*, 1987]. Without loss of generality, assume that the given set of clauses Π_1 contains all the clauses $x_i \vee \neg x_i$ for each x_i in the given alphabet. The set of faults F is equal to the set of literals of the alphabet, and the set of effects M is equal to the set of clauses. The function e is defined as

$$e(f) = \{\ m \mid \text{the clause } m \text{ contains the literal } f\ \}$$

Finally, the set of current effects M' is equal to M, and the number k is equal to the number of atoms in the considered alphabet.

Now, consider two sets of clauses Π_1 and Π_2 over the same alphabet, with $\Pi_1 \subseteq \Pi_2$. The instance $\langle r(\Pi_1), h(\Pi_1)\rangle$ is built as in the previous construction. The instance $\langle r(\Pi_2), h(\Pi_1)\rangle$ has new effects (the clauses in $\Pi_2 \backslash \Pi_1$), and the function e is modified accordingly. However, these new effects need not to be explained (since the set of the current effects is the same of the previous case), thus a diagnosis of size k exists for the second instance if and only if it exists for the first one. □

Note an interesting feature of this kind of proofs: we do not need to prove that the given reduction works, that is, we do not need to prove that a set of clauses is satisfiable if and only if the corresponding diagnosis problem has a solution of size less or equal than k. This can be proved elsewhere (in this case, in [Allemang *et al.*, 1987]). All we have to do is to prove that this reduction is monotonic. This is proved just by comparing two similar instances of the problem of diagnosis (completely disregarding the instances of 3sat).

Of course, not all the proofs of NP hardness use a reduction from 3sat. Giving an appropriate definition of monotonicity of a reduction from the problems *node cover, clique, exact cover, three-exact cover*, one can prove that the existence of a monotonic reduction to a problem implies the nucompNP hardness of it.

Definition 11 *A reduction $\langle r,h\rangle$ from Node Cover (*nc*) or* clique *to a problem of pairs S is said to be monotonic if, for any two sets of edges $E_1 \subseteq E_2$ over the same set of nodes N, it holds*

$$\langle r((N,E_1),k),h((N,E_1),k)\rangle \in S \Leftrightarrow$$
$$\langle r((N,E_2),k),h((N,E_1),k)\rangle \in S$$

The monotonic reductions from nc and clique have the same property of the monotonic reductions from 3sat. Namely, they prove the nucompNP hardness of problems.

Theorem 4 *If there exists a monotonic polynomial reduction from in* nc *or* clique *to a problem of pairs B, then B is* nucompNP *hard (and thus it does not belong to* compP*, unless the polynomial hierarchy collapses).*

The above theorem is useful for reductions whose starting point is a problem on graphs. For problems of sets, we give the definition of monotonic reduction using the problem Exact Cover (ec).

Definition 12 *A reduction $\langle r,h\rangle$ from the problem Exact Cover (*ec*) or Three-Exact Cover (*x3c*) to a problem of pairs B is said to be monotonic if, for any two subsets S_1, S_2 of W, with $S_1 \subseteq S_2$, it holds*

$$\langle r(S_1),h(S_1)\rangle \in B \quad \Leftrightarrow \quad \langle r(S_2),h(S_1)\rangle \in B$$

These monotonic reductions have the same property of the other monotonic reductions.

Theorem 5 *If there exists a monotonic polynomial reduction from* ec *or* x3c *to a problem of pairs B, then B is* nucompNP *hard (and thus it does not belong to* compP*, unless the polynomial hierarchy collapses).*

4 PLANNING

STRIPS [Fikes and Nilsson, 1971] is a formalism for expressing planning problems. In the last years, it has been mainly used for studying the theoretical complexity of planning.

A STRIPS planning problem is a 4-tuple $\langle \mathcal{P},\mathcal{O},\mathcal{I},\mathcal{G}\rangle$, where $\mathcal{P}$ is a set of conditions, $\mathcal{O}$ is the set of operators (actions that can be performed to accomplish the goal), $\mathcal{I}$ is the initial state, and $\mathcal{G}$ is the goal.

The *conditions* are facts that can be true or false in the world of interest. A *state* S is a set of conditions, and represents the state of the world in a certain time point. The conditions in S are those representing facts that are true in the world, while those not in S represent facts currently false.

The *initial state* is a state, thus a set of conditions. The *goal* is specified by giving a set of conditions that should be achieved, and another set specifying which conditions should not be made true. Thus, a goal $\mathcal{G}$ is a pair $\langle \mathcal{M},\mathcal{N}\rangle$, where $\mathcal{M}$ is the set of conditions that should be made true, while $\mathcal{N}$ is the set of conditions that should be made false.

The *operators* are actions that can be performed to achieve the goal. Each operator is a 4-tuple $\langle \phi,\eta,\alpha,\beta\rangle$, where ϕ, η, α, and β are sets of conditions. When executed, such an operator makes the conditions in α true, and those in β false, but only if the conditions in ϕ are currently true and those in η are currently false. The conditions in ϕ and η are called the positive and negative *preconditions* of the operator. The conditions in α and β are called the positive and negative *effects* or *postconditions* of the operator.

Given an instance of a STRIPS planning problem $\langle \mathcal{P},\mathcal{O},\mathcal{I},\mathcal{G}\rangle$, we define a plan for it as a sequence of operators that, when executed in sequence from the initial state, lead to a state where all the conditions in $\mathcal{M}$ are true and all those in $\mathcal{N}$ are false. More details about the definition of STRIPS can be found in [Fikes and Nilsson, 1971] and [Bylander, 1991].

The instances of a STRIPS planning problem have a clear fixed-varying part pattern. The set of conditions and operators are the fixed part of the input, since they represent the general description of the world. It is very likely that we need to solve many planning problems w.r.t. the same world. Consider, for example, a robot that delivers the mail on a floor of a building. After delivered the first mail, it is necessary to find a new plan to deliver the second one, etc. Even if we formalize the problem in such a way the goal is to deliver all the mail of the day, the day after there is a need of a new plan, as the new mail to be delivered is different. This example shows that, given a general description of the world (conditions and operators), it is likely that there are many planning problems that differ only for the initial state and the goal.

As a result, STRIPS planning is a problem for which compilation may be useful. The fixed part of the input is composed of $\mathcal{P}$ and $\mathcal{O}$, where a compilation is allowed, while the varying part is composed of the initial state $\mathcal{I}$ and the goal $\mathcal{G}$.

The problem analyzed here is PLANSAT+: given a STRIPS instance such that all the operators have only positive postconditions, decide whether there exists a plan. This problem has been proved NP complete by Bylander [1991]. We prove that Bylander's proof is monotonic. First of all, a STRIPS problem is expressed as a language of pairs as

$$\text{PLANSAT+} = \{\langle\langle\mathcal{P},\mathcal{O}\rangle,\langle\mathcal{I},\mathcal{G}\rangle\rangle \mid \text{there exists a plan}\}$$

Bylander's reduction is from 3sat. Let Π_1 be a set of clauses, each composed of three literals, over the alphabet X. For each variable in X there are two conditions T_i and F_i. There is also a condition C_j for each clause in Π_1. For each variable there are two operators $\langle\emptyset,\{F_i\},\{T_i\},\emptyset\rangle$ and $\langle\emptyset,\{T_i\},\{F_i\},\emptyset\rangle$. There is also an operator for each literal of each clause. If the literal is x_i in the clause γ_j then the operator is $\langle\{T_i\},\emptyset,\{C_j\},\emptyset\rangle$, otherwise it is $\langle\{F_i\},\emptyset,\{C_j\},\emptyset\rangle$. The initial state is empty, while the goal is $\mathcal{G} = \langle\mathcal{M},\emptyset\rangle$, where $\mathcal{M} = \{C_j \mid \gamma_j \in \Pi\}$. As in the example of diagnosis, there is no need to prove that this reduction works. We trust Bylander.

Given two set of clauses of three literals Π_1 and Π_2 such that $\Pi_1 \subseteq \Pi_2$, the result of $r(\Pi_2)$ differs from $r(\Pi_1)$ only for the fact that there are new conditions C_j corresponding to the clauses $\gamma_j \in \Pi_2 \backslash \Pi_1$. Moreover, there are new operators with C_j as postcondition.

Let us compare $\langle r(\Pi_1), h(\Pi_1)\rangle$ and $\langle r(\Pi_2), h(\Pi_1)\rangle$. The new conditions have no effect on the existence of a plan, since a. they are not precondition of any operator, and b. they are not part of the goal. As a result, if a plan contains some of the new operators, then they can be removed from the plan. This reduced plan still leads to a state in which the goal is satisfied, because the new operators do not affect the preconditions of the old operators, neither they make true or false the conditions of the goal.

This proves that the given reduction is monotonic, thus the problem PLANSAT+ is nucompNP hard, and thus it cannot be compiled into P.

5 REASONING ABOUT ACTIONS

In this paper we consider the language $\mathcal{A}$ which expresses domains of actions. A detailed description of $\mathcal{A}$ is given in [Gelfond and Lifschitz, 1993], where more examples are also provided. For the sake of completeness, we recall here the syntax of $\mathcal{A}$, with a short explanation of the semantics.

There are two nonempty sets of symbols, called fluent names and action names. Fluents are facts that can be true or false. They are called "fluents" because their value may change over time. A fluent expression is a fluent name possibly preceded by the negation symbol $\neg$. If F is a fluent expression and $A_1, \ldots, A_m$ are action names, then

$$F \text{ after } A_1; \ldots; A_m$$

is a value proposition. Such a proposition means that after the execution of the sequence of actions $A_1; \ldots; A_m$ the fluent expression F becomes true. If $m = 0$, the above statement is written

$$\text{initially } F$$

This expresses the value of a fluent in the initial state.

An effect proposition is an expression of the form

$$A \text{ causes } F \text{ if } P_1, \ldots, P_m$$

where A is an action name and $F, P_1, \ldots, P_m$ are fluent expressions. It means that the action A, when performed, makes the fluent expression F true, if the fluent expressions $P_1, \ldots, P_m$ are true. The fluents $P_1, \ldots, P_m$ are called preconditions, while F is the effect or postcondition of the proposition.

A domain description is a set of value and effect propositions.

A model of a domain description is a pair (initial state, transition function), where the transition function is a function that expresses how the states change in response to actions. We do not formally give the semantics of $\mathcal{A}$ here. The problem of interest is the entailment: $D \models V$ holds if and only if all the models of D are also models of V, where D is a domain description and V is a single value proposition.

The problem of entailment $D \models V$ in $\mathcal{A}$ is proved to be coNP complete in [Liberatore, 1997b]. Since a single domain description D may be queried many times w.r.t. several different propositions V, it makes sense to compile D once, if this simplifies the problem. Thus, it is worthwhile to analyze entailment in $\mathcal{A}$ w.r.t. our framework of compilation: the domain description D is the fixed part of the input, while V is the varying part.

For the sake of simplicity, in this paper we only consider NP problems. This does not prevent us from analyzing the problem of entailment in $\mathcal{A}$. Indeed, $D \models F$ after $A_1; \ldots ; A_m$ if and only if $D \cup \{\neg F$ after $A_1; \ldots ; A_m\}$ is not consistent, and consistency in $\mathcal{A}$ is NP complete. As a result, if we can show that the problem of consistency of $D \cup \{V\}$ (where V is a value proposition) is compilable (where D is the fixed part) then the problem of entailment is compilable, and vice versa. Indeed, what we prove is that the problem of deciding the consistency of a domain description with a value proposition is not compilable. In terms of languages of pairs, the problem can be formalized as:

$$\mathcal{A}\text{sat} = \{\langle D, V\rangle \mid D \cup \{V\} \text{ is consistent}\}$$

The reduction that proves NP hard the problem of consistency given in [Liberatore, 1997b] is not monotonic. Consider the two set of clauses $\Pi_1 = \{x\}$ and $\Pi_2 = \{x, \neg x\}$ which are built on the same alphabet. We have

$$\begin{aligned} r(\Pi_1) &= \{\text{ initially } \neg F,\ A \text{ causes } F \text{ if } x\} \\ r(\Pi_2) &= \{\text{ initially } \neg F,\ A \text{ causes } F \text{ if } x, \\ &\qquad A \text{ causes } F \text{ if } \neg x\} \\ h(\Pi_1) &= \neg F \text{ after } A \end{aligned}$$

Thus $r(\Pi_1) \cup \{h(\Pi_1)\}$ is satisfiable while $r(\Pi_2) \cup \{h(\Pi_1)\}$ is not. As a result, the reduction is not monotonic.

We give now a monotonic reduction. Let Π_1 be a set of clauses, each composed by three literals. The functions r and h are defined as follows. For each clause $\gamma_i = l_{i_1} \vee l_{i_2} \vee l_{i_3}$ there is an action A_i.

$$\begin{aligned} r(\Pi_1) &= \{\text{ initially } \neg F\} \cup \\ &\qquad \bigcup_{l_{i_1} \vee l_{i_2} \vee l_{i_3} \in \Pi_1} \{A_i \text{ causes } F \text{ if } \neg l_{i_1}, \neg l_{i_2}, \neg l_{i_3}\} \\ h(\Pi_1) &= \neg F \text{ after } A_1; \ldots ; A_n \\ &\qquad \text{where } \Pi_1 = \{\gamma_1, \ldots, \gamma_n\} \end{aligned}$$

This is indeed a reduction from 3sat to $\mathcal{A}$sat.

Theorem 6 *The set Π_1 is satisfiable if and only if $r(\Pi_1) \cup \{h(\Pi_1)\}$ is consistent.*

We can also prove that the given reduction is monotonic. Let Π_2 be a set of clauses over the same alphabet of Π_1, and such that $\Pi_1 \subseteq \Pi_2$. The function r on Π_2 gives:

$$\begin{aligned} r(\Pi_2) &= \{\text{ initially } \neg F\} \cup \\ &\qquad \bigcup_{\gamma_i = l_{i_1} \vee l_{i_2} \vee l_{i_3} \in \Pi_2} \{A_i \text{ causes } F \text{ if } \neg l_{i_1}, \neg l_{i_2}, \neg l_{i_3}\} \end{aligned}$$

That is, there are some new effect propositions: for each $\gamma_i \in \Pi_2 \backslash \Pi_1$ there is a proposition A_i causes F if $\neg l_{i_1}, \neg l_{i_2}, \neg l_{i_3}$.

However, these new actions do not affect the consistency, since a. there is only one effect proposition for each A_i, and b. there is no value proposition in which A_i appears. In semantical terms, such new effect propositions only modify the transition function but not the initial state. This implies that the consistency is not affected.

In this case, we needed to find a monotonic reduction. However, we did it in two steps: first we found the reduction, and afterwards we proved that it is monotonic. These steps are easier than finding a nucomp reduction.

6 BELIEF REVISION

So far we proved only "negative" results, that is, we showed problems for which a compilation does not give a gain in efficiency. In this section we analyze the problem of iterated belief revision. Due to the lack of space, we omit the definitions and the proofs. The basic model we refer to is that of ordinal conditional functions, as can be found, for example, in [Williams, 1994]. The results presented here can be easily extended to other revision operators [Boutilier, 1993, Nayak, 1994, Spohn, 1988, Nayak and Foo, 1997].

The formalism is the following: there is a current knowledge base K and an ordering of plausibility (the ordinal conditional function) that associates a number to each interpretation. This knowledge base K has to be revised with a sequence of revising formulas[4] $P_1, \ldots, P_m$. The problem we are interested in is to determine whether K, revised according to $P_1, \ldots, P_m$, implies another formula Q. This problem is Δ_2^p complete [Liberatore, 1997a].

The result of the revision may be queried many times w.r.t. several formulas Q. As a result, it is worthwhile

[4]Technically, we assume that the plausibility of each of these formulas is 1.

to compile K and $P_1, \dots, P_m$ if this allows for more efficient algorithms.

We can prove that, given K and $P_1, \dots, P_m$, there exists a polynomial size formula K' such that K, revised with $P_1, \dots, P_m$, implies Q if and only if $K' \models Q$

This is a proof of compcoNP membership of the problem. We can determine K' from K and $P_1, \dots, P_m$, during the compilation, and then the problem reduces to $K' \models Q$, which is in coNP. As a result, the compilation reduces the complexity from Δ_2^p to coNP. Note that it is impossible to do any better, since deciding $\models Q$ is already a coNP complete problem (this argument can be used to prove that the problem is comp-coNP complete).

7 OTHER APPLICATIONS

In this section we show that several problems from various fields cannot be made tractable by preprocessing part of the input.

7.1 STEINER TREE

The Steiner Tree problem is defined as: given a graph G whose edges are labeled with integers, a number k and a set of nodes N', decide whether G has a subtree of weight less or equal than k that contains all the nodes of N'.

This is a classical problem of graph theory, and is used to formalize problems of deciding whether a set of points can be connected without exceeding a certain cost. For example, in network design, one wants to know if a set of nodes in a network will remain connected with a given probability (given the probability of failure of the edges). The graph and the weight of the edges are in general fixed (they depends on the structure of the network), while one may ask about the reliability of the network w.r.t. many possible sets of nodes. Thus, the variable part is the set of nodes that the tree must connect.

7.2 NETWORK FLOW

This problem is defined as: given a graph and a collection of disjoint source-sink pairs $\{(s_1, t_1), \dots, (s_k, t_k)\}$, decide if there exists a set of k disjoint paths, each from s_i to t_i.

This problem deals with the ability of a network to support a given traffic. The set of source-sink pairs represents the set of nodes that want to communicate, and thus they are the varying part of the problem.

7.3 REQUIRED PAIRS

The definition of this problem is: given a graph G, a set of pairs $\{(s_1, t_1), \dots, (s_l, t_l)\}$, two nodes s and t and an integer k, decide if there exist a set of k paths from s to t such that for any pair (s_i, t_i) there is a path containing both s_i and t_i.

This problem comes from program testing. To certify a program, one must test all the execution sequences in the program. Even for small programs, the number of them can be extremely large. A compromise is to test only a subset of the execution sequences. Asking that a pair of nodes are together in at least one path is equivalent to ask for a test set that ensure that two segments of code interact in the correct way (see [Ntafos and Hakimi, 1979] for a more detailed explanation). A given program must be tested many times, thus it would be useful to compile its structure in order to speed up the search of test sets.

7.4 HAMILTONIAN CYCLE

The classical problem of Hamiltonian Cycle is: given a graph, decide if there exists a cycle that contains each node exactly once. Here we consider the problem in which we request that the cycle contains exactly a subset of nodes $N' \subseteq N$.

The problem Hamiltonian Cycle is a formalization of problems in which one must visit a set of points, minimizing the total distance traveled. One can hardly see a fixed and a varying part in this problem. However, the variant in which only a subset of nodes must be reached is a typical example of a problem with a fixed and a varying part: the positions of the points (and thus the underlying graph) is fixed, while the set of points that must visited may change from time to time. An example is that of the traveling salesman who has to visit a subset of the cities, or a postman who has to deliver mail only to a subset of people that live in a city. If this is the case, one can afford a long preprocessing time on the graph, if this make easier the finding of the Hamiltonian Cycle.

7.5 CONJUNCTIVE QUERY

Given a conjunctive query in the relational calculus, and a set of relations, the problem is to decide whether the query is true.

This is the decision problem derived from a problem that occurs in relational database systems: given a query, find the tuples that satisfies it. In practical settings, there is a set of typical queries that occur often, while the specific database is not known in advance. In this case it makes sense to compile the query in order to allow a fast answering.

Note that Theorem 7, proving the incompilability of this problem, is not in contrast to a result well known to databases researchers, that proves that Conjunctive Query can be solved in logarithmic space (thus in polynomial time) if the query is constant. This is a good opportunity to remark the difference between "constant" and "fixed". When we say that part of the input data is constant, we mean that this part will be small, or that we are not interested in analyzing the complexity when the size of it becomes large. Instead, when part of the input is fixed, we mean that it can be arbitrarily large but we can afford a long preprocessing on it, because either it is known in advance, or there will be many input instances with the same fixed part.

7.6 SUBGRAPH ISOMORPHISM

The problem of subgraph isomorphism is defined as: given two graphs G and H, decide if there exists a subgraph of G that is isomorphic to H.

This problem is not compilable, either if the fixed part is the "big" graph, or the "small" one. We conjecture that the graph isomorphism problem (given two graphs, decide if they are isomorphic) is compilable into P.

Theorem 7 *The problems: Steiner Tree, Network Flow, Required Pairs, Sub-Hamiltonian Cycle, Conjunctive Query, and Subgraph Isomorphism are* nu-compNP *hard, and thus they cannot be compiled into* P *(unless the polynomial hierarchy collapses).*

8 CONCLUSIONS

In this paper we have shown how the framework of compilation can be used for many problems of Artificial Intelligence. Indeed, we shown that it is theoretically impossible to reduce the complexity of diagnosis, planning, and reasoning about actions via a precompilation of part of the input. A positive result is the reduction from Δ_2^p to coNP of iterated belief revision.

The negative results are obtained by employing the concept of monotonic polynomial reduction. The technique of monotonic polynomial reductions is useful for proving the non-compilability of problem. We proved the generality of the method by applying it to some problems coming from various fields.

Acknowledgments

Many thanks to Marco Schaerf for revising a previous draft of this paper. Part of the work reported here has been done while the author was visiting AT&T Bell Labs, Murray Hill, NJ. Thanks to Henry Kautz for his support. This work has been partially supported by MURST, ASI, and CNR grants.

References

[Allemang *et al.*, 1987] D. Allemang, M. C. Tanner, T. Bylander, and J. R. Josephson. Computational complexity of hypothesis assembly. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI-87)*, pages 1112–1117, 1987.

[Boutilier, 1993] C. Boutilier. Revision sequences and nested conditionals. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 519–525, 1993.

[Bylander, 1991] T. Bylander. Complexity results for planning. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 274–279, 1991.

[Cadoli *et al.*, 1995] M. Cadoli, F. M. Donini, P. Liberatore, and M. Schaerf. The size of a revised knowledge base. In *Proceedings of the Fourteenth ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS-95)*, pages 151–162, 1995. Extended version available as Technical Report DIS 34-96, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", November 1996.

[Cadoli *et al.*, 1996] M. Cadoli, F. M. Donini, P. Liberatore, and M. Schaerf. Feasibility and unfeasibility of off-line processing. In *Proceedings of the Fourth Israeli Symposium on Theory of Computing and Systems (ISTCS-96)*, pages 100–109. IEEE Computer Society Press, 1996. URL = http://www.dis.uniroma1.it/PUB/AI/papers/cado-etal-96.ps.gz.

[Cadoli *et al.*, 1997] M. Cadoli, F. M. Donini, M. Schaerf, and R. Silvestri. On compact representations of propositional circumscription. *Theoretical Computer Science*, 182:183–202, 1997.

[Chandra and Merlin, 1977] A. Chandra and P. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proceedings of the Fifth ACM Symposium on Theory of Computing (STOC-77)*, pages 77–90, 1977.

[Fikes and Nilsson, 1971] R. Fikes and N. Nilsson. STRIPS: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[Garey and Johnson, 1979] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, San Francisco, Ca, 1979.

[Gelfond and Lifschitz, 1993] M. Gelfond and V. Lifschitz. Representing action and change by logic programs. *Journal of Logic Programming*, 17:301–322, 1993.

[Gogic *et al.*, 1995] G. Gogic, H. A. Kautz, C. Papadimitriou, and B. Selman. The comparative linguistics of knowledge representation. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 862–869, 1995.

[Johnson, 1990] D. S. Johnson. A catalog of complexity classes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A, chapter 2. Elsevier Science Publishers (North-Holland), Amsterdam, 1990.

[Karp and Lipton, 1980] R. M. Karp and R. J. Lipton. Some connections between non-uniform and uniform complexity classes. In *Proceedings of the Twelfth ACM Symposium on Theory of Computing (STOC-80)*, pages 302–309, 1980.

[Karp, 1972] R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. New York: Plenum, 1972.

[Kautz and Selman, 1992] H. A. Kautz and B. Selman. Forming concepts for fast inference. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 786–793, 1992.

[Liberatore, 1997a] P. Liberatore. The complexity of iterated belief revision. In *Proceedings of the Sixth International Conference on Database Theory (ICDT-97)*, pages 276–290, 1997.

[Liberatore, 1997b] P. Liberatore. The complexity of the language $\mathcal{A}$. *Electronic Transactions on Artificial Intelligence*, 1(1–3):13–37, 1997. Available at http://www.ep.liu.se/ea/cis/1997/006/.

[Nayak and Foo, 1997] A. Nayak and N. Foo. Reasoning without minimality. In *IJCAI-97 Workshop on Nonmonotonic Reasoning, Action and Change*, pages 127–138, 1997.

[Nayak, 1994] A. Nayak. Iterated belief change based on epistemic entrenchment. *Erkenntnis*, 41:353–390, 1994.

[Ntafos and Hakimi, 1979] S. Ntafos and S. Hakimi. On path cover problems in digraphs and applications to program testing. *IEEE Transaction on Software Engigneering*, SE-5:520–529, 1979.

[Peng and Reggia, 1986] Y. Peng and J. Reggia. Plausibility of diagnostic hypothesis. In *Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86)*, pages 140–145, 1986.

[Preparata and Shamos, 1985] F. P. Preparata and M. I. Shamos. *Computational Geometry: An introduction.* Springer-Verlag, 1985.

[Spohn, 1988] W. Spohn. Ordinal conditional functions: A dynamic theory of epistemic states. In *Causation in Decision, Belief Change, and Statistics*, pages 105–134. Kluwer Academics, 1988.

[Williams, 1994] M. Williams. Transmutations of knowledge systems. In *Proceedings of the Fourth International Conference on the Principles of Knowledge Representation and Reasoning (KR-94)*, pages 619–629, 1994.

A PROOFS

In this appendix we report the proofs of two of the theorems stated in the text. The first is Theorem 2, which state that the existence of a monotonic reduction from 3sat to a problem implies its nucompNP hardness. Then, we prove the nucompNP hardness of the Steiner Tree problem.

Theorem 2 *If there exists a monotonic polynomial reduction from* 3sat *to a problem of pairs S, then S is* nucompNP *hard.*

Proof. Suppose there exists a monotonic polynomial reduction $\langle r, h\rangle$ from 3sat to B. We prove that there exists a nu-comp reduction f_1, f_2, g from the problem *3sat (which is compNP hard: see [Cadoli *et al.*, 1996]) to B.

Let $||\Pi||$ denote the size of Π (that is, the size of the string used to represent it), and $Var(\Pi)$ be the number of atoms in it. Furthermore Π_m is the set of all the clauses of three literals over a set of variables $\{x_1, \ldots, x_m\}$.

Let f_1, f_2, g be defined as follows

$$\begin{aligned} f_1(s,m) &= r(\Pi_m) \\ f_2(s,m) &= \epsilon \end{aligned}$$

$$g(a,\Pi) = h(\Pi \cup \{x_{Var(\Pi)+1} \vee \neg x_{Var(\Pi)+1}, \ldots, x_{||\Pi||} \vee \neg x_{||\Pi||}\})$$

Let $\Pi' = \Pi \cup \{x_{Var(\Pi)+1} \vee \neg x_{Var(\Pi)+1}, \ldots, x_{||\Pi||} \vee \neg x_{||\Pi||}\}$. Using the fact that $\langle r, h\rangle$ is a monotonic polynomial reduction we have that

$$\begin{array}{lll} \langle x,\Pi\rangle \in *\text{3sat} & \text{iff} & \Pi \text{ is satisfiable} \\ & \text{iff} & \Pi' \text{ is satisfiable} \\ & \text{iff} & \langle r(\Pi'), h(\Pi')\rangle \in B \\ & \text{iff} & \langle r(\Pi_{Var(\Pi')}), h(\Pi')\rangle \in B \\ & \text{iff} & \langle r(\Pi_{||\Pi||}), h(\Pi')\rangle \in B \end{array}$$

But $r(\Pi_{||\Pi||}) = f_1(x, ||\Pi||)$ and $h(\Pi') = g(a,\Pi)$, thus $\langle x,\Pi\rangle \in *$3sat if and only if $\langle f_1(x,||\Pi||), g(f_2(x,||\Pi||),\Pi)\rangle \in B$. □

Theorem 8 *Steiner Tree (*st*) is* nucompNP *hard.*

Proof. The reduction given by Karp in A.D. 1972 is not monotonic. The proof is by means of a reduction from Node Cover. The problem is that the instance of the Node Cover problem must be a graph with labels on edges, and this is not allowed by Theorem 4. However, with few changes the proof can be made monotonic.

Let $S = \{S_1, \ldots, S_m\}$ be a generic instance of the problem x3c. Let $\bigcup S = \{u_1, \ldots, u_n\}$. The corresponding instance of the problem st is

$$\begin{array}{lll} N & = & \{n_0\} \cup S \cup \{\langle u_i, S_j\rangle \mid u_i \in S_j\} \\ E & = & \{\langle n_0, S_j\rangle\} \cup \{\langle S_j, \langle u_i, S_j\rangle\rangle\} \cup \\ & & \{\langle\langle u_i, S_j\rangle, \langle u_i, S_z\rangle\rangle\} \\ w & : & w(\langle n_0, S_j\rangle) = 3ln \\ & & w(\langle S_j, \langle u_i, S_j\rangle\rangle) = 0 \\ & & w(\langle\langle u_i, S_j\rangle, \langle u_i, S_z\rangle\rangle) = 1 \\ N' & = & \{n_0\} \cup \{\langle u_i, S_j\rangle\} \\ k & = & ln^2 + \sum(p(u_i) - 1) \end{array}$$

where l is the number of all the possible subsets of three elements of a set of m elements, and $p(u_i)$ is the number of sets in which u_i is present.

Suppose there exists a subset $S' \subseteq S$ such that $\bigcup S' = \bigcup S$ and $S_j \cap S_z = \emptyset$ for each $S_j, S_z \in S'$. A Steiner tree in the graph above is the following.

$$\begin{array}{lll} T & = & \{\langle n_o, S_j\rangle \mid S_j \in S'\} \cup \{\langle S_j, \langle u_i, S_j\rangle\rangle \mid S_j \in S'\} \\ & & \cup\{\langle\langle u_i, S_j\rangle, \langle u_i, S_z\rangle\rangle \mid S_j \in S'\} \end{array}$$

It is easy to see that this tree has weight

$$3ln \cdot n/3 + \sum(p(u_i) - 1) = k$$

Suppose there exists a Steiner tree T of weight less or equal than k. We define

$$S' = \{S_j \mid \langle n_0, S_j\rangle \in T\}$$

First of all, we have $|S'| \leq n/3$: otherwise, the weight of the tree would be greater than k, as one can easily prove.

We have also $\bigcup S' = \bigcup S$. The node n_0 is the root of the tree (we recall that the graph is direct). Given an element u_i, each node $\langle u_i, S_j\rangle$ must be reached by the tree, thus it must be the case that

$$\begin{array}{ll} & \langle n_0, S_j\rangle, \langle S_j, \langle u_i, S_j\rangle\rangle \in T \\ \text{or} & \langle n_0, S_z\rangle, \langle S_z, \langle u_i, S_z\rangle\rangle, \langle\langle u_i, S_z\rangle, \langle u_i, S_y\rangle\rangle, \ldots, \\ & \langle\langle u_i, S_x\rangle, \langle u_i, S_j\rangle\rangle \in T \end{array}$$

In the first case, $u_i \in S_j$ and $S_j \in S'$. In the second one, $u_i \in S_z$ and $S_z \in S'$. Thus in both cases $u_i \in \bigcup S'$.

Now, since $|S'| \leq n/3$ and $\bigcup S' = \bigcup S$, it must be also that the sets in S' are mutually disjoint.

The monotonicity of this reduction is not trivial to prove. Let S_2 be a set of sets of elements in $\{u_1, \ldots, u_n\}$ such that $S \subseteq S_2$. The instance $\langle r(S), h(S)\rangle$ is that of the previous construction. The instance $\langle r(S_2), h(S)\rangle$ has some new nodes S_z and $\langle u_i, S_z\rangle$ for each $S_z \in S_2 \backslash S$ and $u_i \in S_z$, and the corresponding edges. Since the weight of the tree and the set of nodes to reach are the same, it follows that a Steiner tree for the instance $\langle r(S), h(S)\rangle$ is also a Steiner tree for $\langle r(S_2), h(S)\rangle$.

The hard part of the proof is to show that if the instance $\langle r(S_2), h(S)\rangle$ has a Steiner tree of weight at most k, then the same holds for the instance $\langle r(S), h(S)\rangle$. This is proved by showing that the Steiner tree for $\langle r(S_2), h(S)\rangle$ does not contain the new nodes. Informally, the Steiner tree for $\langle r(S_2), h(S)\rangle$ may have at most $n/3$ nodes of the type S_z. Suppose that one of these nodes corresponds to a set $S_z \in S_2 \backslash S$. Since the tree must also contain all the nodes $\langle u_i, S_j\rangle$, it follows that each node $\langle u_i, S_z\rangle$ must be linked to all the corresponding nodes $\langle u_i, S_j\rangle$. This makes a total weight of $p(u_i)$. As a result, the total weight of the tree is greater than k. □

Compiling Devices: A Structure-Based Approach

Adnan Darwiche
Department of Mathematics
American University of Beirut
PO Box 11 - 236
Beirut, Lebanon
darwiche@aub.edu.lb

Abstract

We present an approach for compiling a device into a propositional sentence (in negation normal form) and then show how to answer interesting diagnostic queries based on such a device compilation. We present two key results about our compilation approach, which makes it of interest to the diagnostic community. First, the complexity of answering diagnostic queries — which includes finding minimum-cardinality diagnoses — is linear in the size of the device compilation. Second, and more importantly, the size of the device compilation depends on the device structure in a precise way. In particular, if the device structure contains no cycles, the compilation is shown to be linear in the size of the device. In general though, the size of the device compilation is shown to be exponential only in w^*: the width of the graph representing the device structure. Therefore, our approach seems to be among the first model-based approaches for compiling devices that offer formal guarantees on the size of the device compilation.

1 Introduction

Compiling devices for the purpose of answering diagnostic queries has been receiving continued interest in the model-based community. The basic idea here is to divide computational work into two phases: off-line and on-line. In the off-line phase, one generates a compiled representation of the device and then uses that same representation in the on-line phase to answer multiple diagnostic queries. For such a compilation to be useful, however, it is typically expected that the computational complexity of answering queries in the on-line phase is low — say, linear in the size of the compiled representation — and that the overhead involved in such computation is also relatively low.

Although a well-accepted theory exists for model-based diagnosis [12, 4], no such theory seems to exist for compiling devices. Most proposals for compilation are ad-hoc techniques for enhancing the performance of various model-based diagnosis systems. An exception to this, however, seems to be the compilation approach proposed by de Kleer [6]. According to this approach, a device is modeled using a set of logical sentences and is compiled in an off-line phase by computing the prime implicates of the sentences constituting the device model. Given such implicates, one can generate in the on-line phase all the minimal conflicts corresponding to any given observation by simply traversing the device implicates.

This approach is marked by its elegance, but it has proven not to be practical for two reasons. First, computing conflicts is only a first step in answering diagnostic queries and it remains to be seen whether further steps can be accomplished in linear time. Second, the compiled representation itself (the device prime implicates) tends to be exponential, therefore defeating the purpose of compilation. In addition, there does not seem to be a good way of predicting whether the compiled device will be of a manageable size or not. This in turn calls for a theory that can predict the size of the compiled device in terms of some device properties.

Our goal in this paper is to address this particular problem. Specifically, we present a new compilation approach which is marked by two features: (1) the simplicity of its on-line phase which can be performed in time linear in the size of the compiled representation; and (2) the formal guarantees it offers on the computational complexity of the off-line phase, namely, on the size of the compiled device in particular.

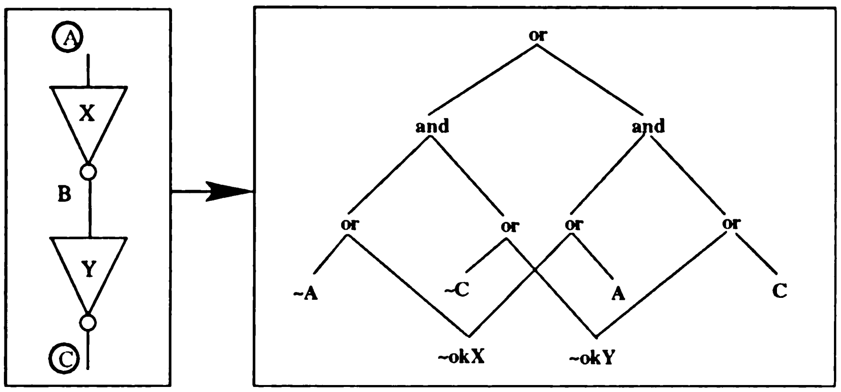


Figure 1: On the left, a device with two observables A and C — that is, variables about which we plan to collect observations. On the right, a compiled system description that can be used to answer diagnostic queries about the collected observations.

According to our approach, a device is compiled into a propositional sentence in ***negation normal form*** [1]; therefore, the guarantees we offer are on the time to generate, and the space to store, such a propositional sentence. The device itself is represented using a classical system description — that is, a set of propositional sentences [12, 4] — and the result of compilation is referred to as a ***compiled system description (CSD).*** An example device and its compiled system description are shown in Figure 1. We will discuss the structure and properties of compiled system descriptions at length in Section 3, after having motivated further the process of device compilation in Section 2. We then turn in Section 4 to the on-line phase of our compilation approach where we discuss linear time operations for answering diagnostic queries. Section 5 is dedicated to the off-line phase of our approach: We discuss the compilation algorithm (which is based on a reduction to computing logical consequences [3]) and present our guarantees:

1. If the device structure contains no cycles, then the compiled system description can be generated in linear time and can be stored in linear space.[1]

2. More generally, the time to generate a compiled system description, and the space to store it, are shown to be exponential only in w^*: the width of the graph representing the device structure.

This means that compiling devices for the purpose of answering diagnostic queries can be achieved under the same guarantees that one finds in computations based on belief and constraint networks which are popular in the probabilistic and constraint-satisfaction communities.

[1]Assuming the number of inputs per component is a constant.

2 Compiling a Device

Consider the simple device shown in Figure 1 which has one input A and one output C. The circles enclosing A and C declare these variables as being observables; that is, device variables about which we plan to collect observations. Given that each one of these variables could be either on or off, we have a total of four possible device observations: $A \wedge C$, $A \wedge \neg C$, $\neg A \wedge C$ and $\neg A \wedge \neg C$. Our task: compute the set of minimum-cardinality diagnoses given any of these observations. Informally, a diagnosis is a sentence of the form $[\neg]okX \wedge [\neg]okY$, which is consistent with the device model and the given observation. Here, $[\neg]$ means that the negation sign is optional. A minimum-cardinality diagnosis is a diagnosis which has a minimum number of negation signs (faults). The formal definitions of a device model, a diagnosis, and a minimum-cardinality diagnosis will follow later.

If we were to solve this problem using our compilation approach, we would proceed as follows. First, we would model the device using a set of propositional sentences:

$$\Delta = \left\{ \begin{array}{c} okX \supset A \equiv \neg B, \\ okY \supset B \equiv \neg C \end{array} \right\}.$$

We would then pass this model Δ and the set of observables A and C to our diagnostic compiler which would generate, in an off-line phase, the propositional sentence shown in Figure 1. This sentence alone, as we shall show, is sufficient to answer all queries of the form:

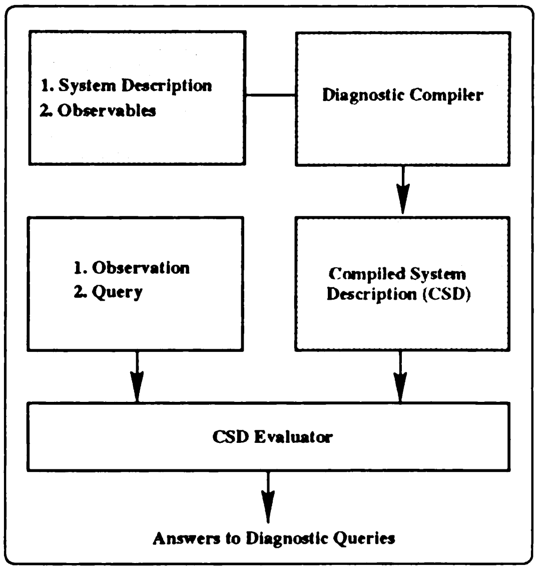


Figure 2: The architecture of the proposed diagnostic compiler.

1. What is the minimum number of possible faults given some observation about A and C?
2. What is the minimum-cardinality diagnoses in such a case?

For example, if the observation is $A \land \neg C$, then the minimum number of faults is 1 and the minimum-cardinality diagnoses are $okX \land \neg okY$ and $\neg okX \land okY$.

The architecture of our proposed compilation approach is shown in Figure 2. The shaded boxes are those involved in the off-line phase: Given a system description and a set of observables, output a CSD. Once we have a CSD and a system observation, the CSD evaluator can then be invoked to answer the necessary diagnostic query in linear time. The CSD evaluator will be discussed in Section 4. In the next section, we discuss the structure and properties of the propositional sentence representing a CSD.

3 Compiled System Descriptions

In the rest of this paper, a system description refers to a set of propositional sentences Δ which are constructed from two disjoint sets of atoms: The *assumables,* denoted $\mathbf{A}$, and the *non-assumables,* denoted $\mathbf{P}$. Intuitively, assumables are atoms that represent the health of components while non-assumables are all remaining atoms. For the device in Figure 1, the assumables are okX and okY and the non-assumables are A, B and C. *System observables* are a subset of the non-assumables and are denoted by $\mathbf{O}$.

If $\mathbf{X}$ is a set of atoms, then an $\mathbf{X}$-*sentence* is one which is constructed from atoms in $\mathbf{X}$ only, and an $\mathbf{X}$-*instantiation* is a conjunction of literals which contains exactly one literal for each atom in $\mathbf{X}$. The *cardinality* of an instantiation β, denoted $Card(\beta)$, is the number of negative literals appearing in β.

Given the notion of an instantiation, a *candidate diagnosis* can then be defined as an $\mathbf{A}$-instantiation, and a *system observation* can be defined as an $\mathbf{O}$-instantiation.

The diagnoses corresponding to a system description Δ and a system observation α are all candidate diagnoses that are consistent with $\Delta \cup \{\alpha\}$ [12, 4]. We denote such a set by $Dg(\Delta \cup \{\alpha\})$. In general, we will use $Dg(\beta)$ to denote the set of candidate diagnoses consistent with any sentence β.

A main objective of diagnostic reasoning is to obtain information about the set $Dg(\Delta \cup \{\alpha\})$. In this paper, we focus on two pieces of information:

1. The minimum cardinality of diagnoses in the set $Dg(\Delta \cup \{\alpha\})$; and
2. the set of diagnoses in $Dg(\Delta \cup \{\alpha\})$ that have a minimum cardinality.

Our goal is to obtain this information by operating on a device compilation in linear time.

We are now ready to provide a formal definition of a device compilation:

Definition 1 (Compiled System Description) *Given a system description Δ, assumables $\mathbf{A}$ and observables $\mathbf{O}$, a compiled system description, denoted $CSD(\Delta, \mathbf{A}, \mathbf{O})$, is an $(\mathbf{A} \cup \mathbf{O})$-sentence satisfying the following properties:*

1. *$\Delta \models CSD(\Delta, \mathbf{A}, \mathbf{O})$; and*
2. *for any $(\mathbf{A} \cup \mathbf{O})$-sentence γ, $\Delta \models \gamma$ only if $CSD(\Delta, \mathbf{A}, \mathbf{O}) \models \gamma$.*

That is, a compiled system description is a sentence which encodes all the information that the system description encodes about the assumables and observables — nothing more, nothing less! A key property of a compiled system description is that it only mentions assumables and observables. That is, no atom in $\mathbf{P} \setminus \mathbf{O}$ can appear in the compiled system description $CSD(\Delta, \mathbf{A}, \mathbf{O})$.

As it turns out, a compiled system description is all that is needed to compute the set of diagnoses and, therefore, to answer the necessary diagnostic queries:

Theorem 1 *$Dg(\Delta \wedge \{\alpha\}) = Dg(CSD(\Delta, \mathbf{A}, \mathbf{O}) \wedge \alpha)$ for any $\mathbf{O}$-instantiation α.*

In order to ensure that we can answer diagnostic queries in linear time, however, we will insist on generating compiled system descriptions in a specific form known as decomposable negation normal form [1].

Definition 2 *A sentence γ is in negation normal form (NNF) iff the negation operator appears only next to atoms in γ. Moreover, a negation normal form γ is decomposable iff no atoms are shared between any conjuncts that appear in γ.*

Figure 1 depicts graphically a sentence in decomposable negation normal form: $((\neg A \vee \neg okX) \wedge (\neg C \vee \neg okY)) \vee ((\neg okX \vee A) \wedge (\neg okY \vee C))$. Note that the negation operator appears only next to atoms in the sentence. Moreover, there are no common atoms between the conjuncts $\neg A \vee \neg okX$ and $\neg C \vee \neg okY$, neither there are any common atoms between the conjuncts $\neg okX \vee A$ and $\neg okY \vee C$.[2]

We will typically depict negation normal forms using rooted, directed graphs as shown in Figure 1 (the children of a node are shown below it in the graph). Each non-leaf node in the graph represents a conjunction or a disjunction of its children and each leaf node represents a literal.

4 Using CSDs to Answer Diagnostic Queries

We now turn to the process of answering diagnostic queries based on a compiled system description in decomposable NNF, which is the task of the CSD evaluator in Figure 2. The evaluator performs three operations: conditioning, cardinality-computation, and diagnosis-enumeration. The first operation conditions the CSD on a given system observation and is discussed in Section 4.1. The second operation must be performed if we want to compute the minimum number of faults; it requires the first operation and is discussed in Section 4.2. The third step must be performed if we want to compute the minimum-cardinality diagnoses; it requires the first two steps and is discussed in Section 4.3.

[2]To appreciate the computational value of the decomposability property, consider testing satisfiability as an example. It is easy to verify that such a test can be performed in time which is linear in the size of an NNF if the NNF is decomposable. In particular:

1. if α is a literal, then α is satisfiable;
2. if $\alpha = \alpha_1 \vee \ldots \vee \alpha_n$, then α is satisfiable iff some α_i is satisfiable;
3. if $\alpha = \alpha_1 \wedge \ldots \wedge \alpha_n$, then α is satisfiable iff every α_i is satisfiable.

Case 3 above does not hold in general since each of two sentences may be satisfiable but their conjunction may not. However, if the two sentences do not share any atoms, then their satisfiability is enough to guarantee the satisfiability of their conjunction.

4.1 Conditioning

Conditioning is the process of eliminating any reference to observables in a compiled system description while preserving all the information it encodes about the assumables.

Definition 3 (Conditioning) *Let γ be an $(\mathbf{A} \cup \mathbf{O})$-sentence and let α be an $\mathbf{O}$-instantiation. The conditioning of γ on α, written $\gamma \mid \alpha$, is obtained as follows. For each atom O in $\mathbf{O}$:*

- *if O appears positive in α, replace every occurrence of O in γ with true;*
- *otherwise, replace every occurrence of O in γ with false.*

Figure 3 depicts two possible conditionings of the sentence in Figure 1 (we have converted $\neg$*false* to *true* and $\neg$*true* to *false* in this figure). Conditioning on $A \wedge \neg C$ leads to the sentence in Figure 3(a), which is equivalent to $\neg okX \vee \neg okY$. Conditioning on $A \wedge C$ leads to the sentence in Figure 3(b), which is equivalent to *true*. The properties of conditioning are best illustrated by the following theorem:

Theorem 2 *Let γ be an $(\mathbf{A} \cup \mathbf{O})$-sentence and let α be an $\mathbf{O}$-instantiation. Then*

1. *$\gamma \mid \alpha$ is an $\mathbf{A}$-sentence;*
2. *$\gamma \wedge \alpha$ entails $\gamma \mid \alpha$; and*
3. *for any $\mathbf{A}$-sentence β, $\gamma \wedge \alpha \models \beta$ only if $\gamma \mid \alpha \models \beta$.*

That is, by conditioning a compiled system description γ on an observation α we are eliminating any reference to the observables from γ while maintaining all the information that γ encodes about the assumables. This leads to the following theorem:

Theorem 3 *$Dg(CSD(\Delta, \mathbf{A}, \mathbf{O}) \mid \alpha) = Dg(\Delta \cup \{\alpha\})$ for every system observation α.*

That is, by conditioning a compiled system description on an observation we generate a propositional sentence

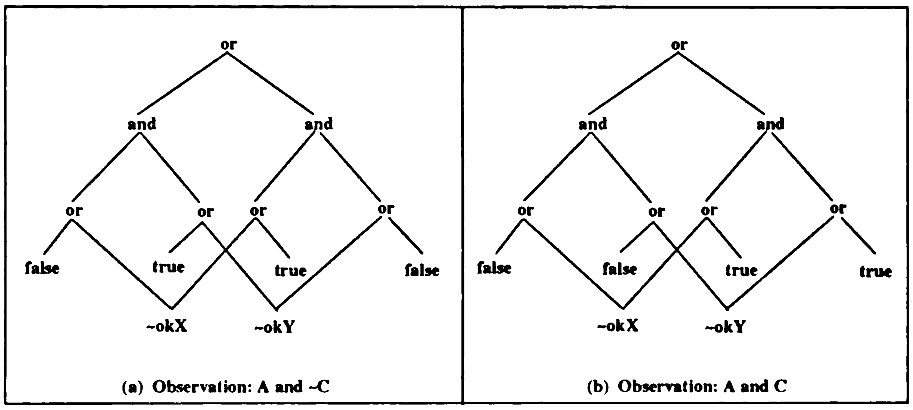


Figure 3: Conditioning a compiled system description.

that mentions assumables only and that characterizes all the diagnoses corresponding to the given system description and observation. Therefore, the sentence contains all the information needed to answer diagnostic queries.

Consider the compiled system description in Figure 1 and its conditioning on two possible observations shown in Figure 3. The sentence in Figure 3(a) results from conditioning on $A \wedge \neg C$ and is equivalent to $\neg okX \vee \neg okY$. Moreover, $\neg okX \vee \neg okY$ and $\Delta \cup \{A \wedge \neg C\}$ characterize the same set of diagnoses: $okX \wedge \neg okY, \neg okX \wedge okY$ and $\neg okX \wedge \neg okY$.

To summarize, the first step performed by a CSD evaluator when answering a query about some observation α is to condition the compiled system description $CSD(\Delta, \mathbf{A}, \mathbf{O})$ on α, leading to the negation normal form $CSD(\Delta, \mathbf{A}, \mathbf{O}) \mid \alpha$. If each observable in $\mathbf{O}$ occurs only once in the compiled system description $CSD(\Delta, \mathbf{A}, \mathbf{O})$, then conditioning $CSD(\Delta, \mathbf{A}, \mathbf{O})$ on α can be done in time which is linear in the size of observation α. In the worst case, however, this conditioning can be done in time which is linear in the size of the compiled system description $CSD(\Delta, \mathbf{A}, \mathbf{O})$.

4.2 Cardinality Computation

Consider the set of diagnoses $Dg(\gamma \mid \alpha)$ characterized by a compiled system description γ that is conditioned on observation α. Our goal in this section is to compute the minimum cardinality of $\gamma \mid \alpha$:

$$mCard(\gamma \mid \alpha) =_{def} \min_{\beta \in Dg(\gamma|\alpha)} Card(\beta).$$

This cardinality is nothing but the minimum number of possible faults given that we have observed α. Consider the compiled system description γ in Figure 1 and the observation $\alpha = A \wedge \neg C$. The conditioned sentence $\gamma \mid \alpha$ is given in Figure 3(a) and it characterizes three diagnoses $okX \wedge \neg okY$, $\neg okX \wedge okY$ and $\neg okX \wedge \neg okY$, which have cardinalities 1, 1 and 2, respectively. Therefore, the minimum cardinality of $\gamma \mid \alpha$ is 1, which is also the minimum number of possible faults given the observation α.

Computing the minimum cardinality of a sentence can be performed in linear time if the sentence is in decomposable negation normal form:

Theorem 4 *Given that $\gamma_1 \vee \gamma_2$ and $\gamma_1 \wedge \gamma_2$ are in decomposable NNF:*

1. $mCard(\gamma) = \begin{cases} 0, & \textit{if } \gamma \textit{ is a positive literal;} \\ 1, & \textit{if } \gamma \textit{ is a negative literal.} \end{cases}$
2. $mCard(\gamma_1 \vee \gamma_2) = \min(mCard(\gamma_1), mCard(\gamma_2))$.
3. $mCard(\gamma_1 \wedge \gamma_2) = mCard(\gamma_1) + mCard(\gamma_2)$. ∎

All of the above cases are straightforward, except possibly for the last one which follows because decomposability ensures that γ_1 and γ_2 share no atoms. Figure 4 illustrates this computation for the sentences in Figure 3. The proof of this theorem is given in [3], where pseudocode is also given for computing cardinalities of nodes in a decomposable negation normal form that is represented as a rooted, directed graph.

Armed with conditioning and cardinality-computation, we can now answer in linear time any query of the form: What is the minimum number of possible faults given a system observation α? To answer this query, we simply condition the compiled system description γ on the observation α, leading to $\gamma \mid \alpha$, and then compute $mCard(\gamma \mid \alpha)$.

The previous query is important for at least two reasons:

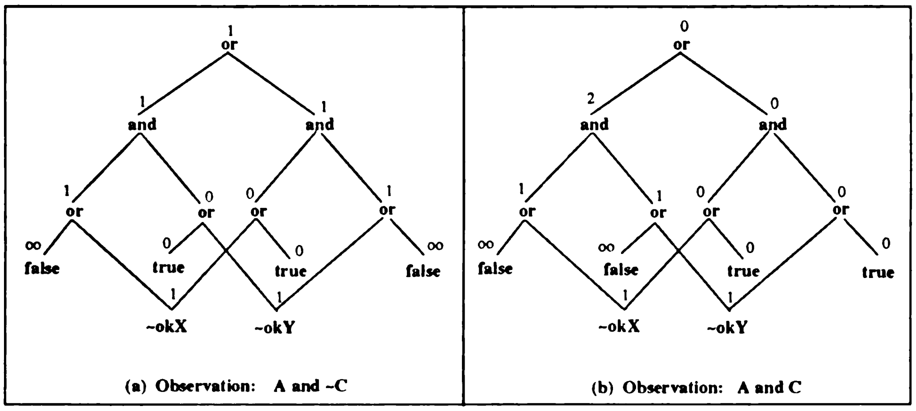


Figure 4: Computing minimum cardinality.

1. It is required for computing minimum-cardinality diagnoses as discussed in the following section.
2. It can be used to decide whether the system is exhibiting an abnormal behavior. Specifically, if $mCard(\gamma \mid \alpha) > 0$, then we know that there is at least one fault and, hence, the observation α is inconsistent with the system normal behavior.

If $mCard(\gamma \mid \alpha) = 0$, however, then we know that the observation α is consistent with the system normal behavior. Consider Figure 4(b) where $mCard(\gamma \mid \alpha) = 0$. This zero cardinality implies that the candidate diagnosis $okX \wedge okY$ is consistent with the system description and observation since this is the only candidate diagnosis with zero cardinality. Therefore, the observation $A \wedge C$ in this case does not imply a system failure.

4.3 Diagnosis Enumeration

If $mCard(\gamma \mid \alpha) > 0$, we know that we have at least one fault. Our goal in such a case is to enumerate all diagnoses that have a minimum number of faults, which we denote by $mDg(\gamma \mid \alpha)$. The formal definition of this set of diagnoses is:

$$\{\beta : \beta \in Dg(\gamma \mid \alpha) \text{ and } Card(\beta) = mCard(\gamma \mid \alpha)\}.$$

This set of diagnoses can be computed quite efficiently given that the sentence $\gamma \mid \alpha$ is in decomposable negation normal form. Before we show how this can be done, however, we need to introduce some definitions:

- Given a sentence γ, $Atoms(\gamma)$ denotes the atoms that appear in γ.
- Given a sentence γ which contains atoms A, $mInst(\gamma)$ denotes the set of minimum cardinality A-instantiations that are consistent with γ.

  For example, if γ is $a \supset b$, then there are three instantiations of $\{a, b\}$ that are consistent with γ, but only one of them, $a \wedge b$, has the minimum cardinality. Therefore, $mInst(\gamma) = \{a \wedge b\}$ in this case.
- Given a set of instantiations I and a set of atoms A, $Extend(I, A)$ denotes the set which results from replacing each instantiation β in I with $\beta \wedge \beta'$ where β' is the conjunction of all atoms in A that do not appear in β.

  For example, if $I = \{\neg okX \wedge okZ, \neg okY \wedge okZ\}$ and $A = \{okX, okY, okZ\}$, then $Extend(I, A)$ equals $\{\neg okX \wedge okZ \wedge okY, \neg okY \wedge okZ \wedge okX\}$.

Given these definitions, we are now ready for enumerating the minimum cardinality diagnoses. This is given by the following theorem:

Theorem 5 *Given that $\gamma_1 \vee \gamma_2$ and $\gamma_1 \wedge \gamma_2$ are in decomposable NNF:*

1. *$mInst(\gamma) = \{\gamma\}$ if γ is a literal.*
2. *If $\gamma = \bigwedge_i \gamma_i$, then $mInst(\gamma) =$*
$$\left\{ \bigwedge_i \beta_i : \beta_i \in mInst(\gamma_i) \right\}.$$
3. *If $\gamma = \bigvee_i \gamma_i$, then $mInst(\gamma) =$*
$$\bigcup_{mCard(\gamma_i) = mCard(\gamma)} Extend(mInst(\gamma_i), Atoms(\gamma)).$$

Moreover, $mDg(\gamma) = Extend(mInst(\gamma), \mathbf{A})$. ∎

According to this theorem, we can enumerate minimum cardinality diagnoses using a recursive procedure which visits each node of the decomposable NNF only once. The enumeration process is similar to that of computing disjunctive normal forms, with the following exceptions:

1. The consistency of $\bigwedge_i \beta_i$ is not checked in Case 2.

   Intuition: Decomposability ensures that no atoms are shared between the conjuncts β_i, therefore, $\bigwedge_i \beta_i$ must be consistent.

2. A disjunct is ignored in Case 3 if its cardinality is higher than the cardinality of one of its siblings.

   Intuition: Disjuncts that do not have a minimum cardinality will contribute terms that will only be included in diagnoses that are not minimal; therefore, these terms can be safely ignored.

3. After we enumerate the minimum cardinality instantiations of a disjunct γ_i in Case 3, and before we union them with the instantiations obtained from its siblings, we extend them so they mention every atom that appears in γ.

   Intuition: Each instantiation in $mInst(\gamma)$ must mention all the atoms appearing in γ.[3]

We provide in [3] pseudocode for an algorithm that enumerates minimum cardinality diagnoses based on Theorem 5. For computing $mDg(\gamma)$, the complexity of the algorithm is shown to be linear in the size of the decomposable negation normal form γ and quadratic in the number of minimal diagnoses in $mDg(\gamma)$.

5 Generating Compiled System Descriptions

In the previous section, we provided algorithms for answering key diagnostic queries based on a compiled system description. The algorithms are linear in the size of a compiled system description which means that on-line diagnostic reasoning can be very efficient if one can generate reasonably-sized compiled system descriptions.

We now turn to the process of generating a compiled system description. That is, given a system description Δ, assumables **A** and observables **O**, we show how to generate a sentence in decomposable negation normal form that satisfies the conditions stated in Definition 1. Once we obtain such a sentence, we can then use the CSD evaluator discussed in the previous section to answer diagnostic queries in linear time.

[3]Note that a term β in a disjunctive normal form of γ does not necessarily mention all atoms appearing in γ.

We will generate a compiled system description by reducing the problem to that of generating a logical consequence.

Definition 4 (Consequence[3]) *Let Δ be a system description, **A** be the assumables and α be a sentence that does not mention assumables. The consequence of α with respect to Δ and **A**, denoted $Cons^{\Delta}_{\mathbf{A}}(\alpha)$, is an **A**-sentence satisfying the following properties:*

1. $\Delta \cup \{\alpha\} \models Cons^{\Delta}_{\mathbf{A}}(\alpha)$; *and*
2. *for any **A**-sentence β, $\Delta \cup \{\alpha\} \models \beta$ only if $Cons^{\Delta}_{\mathbf{A}}(\alpha) \models \beta$.*

Intuitively, the consequence of α with respect to Δ and **A** is the logically strongest **A**-sentence which can be concluded from Δ and α. In the context of diagnosis, the consequence of α is the logically strongest conclusion we can draw about the health of the system given that we have observed α about the system.

We have presented an algorithm for computing logical consequences in [3], which is guaranteed to generate consequences in decomposable negation normal form. As it turns out, the computation of a compiled system description can be reduced to that of computing a logical consequence as the following theorem shows:

Theorem 6 (Compilation)
$CSD(\Delta, \mathbf{A}, \mathbf{O}) \equiv Cons^{\Delta}_{\mathbf{A}\cup\mathbf{O}}(true)$. ∎

This theorem follows immediately from Definitions 1 and 4. It basically says that we can compile the system description Δ with respect to assumables **A** and observables **O** by computing the logically strongest $(\mathbf{A} \cup \mathbf{O})$-sentence entailed by Δ.

An implication of this theorem is that the computational complexity guarantees that apply to consequences also apply to compiled system descriptions.

The time and space complexity of computing consequences is shown in [3] to be exponential only in w^*: the width of the system structure. To explain this guarantee further, we need to explain what we mean by a system structure and, consequently, what we mean by a structured system description.

A *system structure* is a directed acyclic graph which is induced by including a node for each variable in the system, and then extending an arc from variable X to variable Y iff X represents the input of a component whose output is Y. Figure 5 contains an example device with the graph representing its structure.

A *component description* is a set of propositional sentences that describe the behavior of a component. A

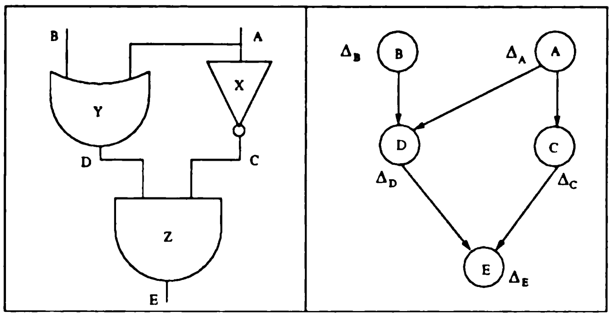


Figure 5: A device with its structure represented using a directed graph.

component description must satisfy some formal conditions that are stated in [3], which are meant to ensure that the description does nothing but describe the component. For example, the only atoms appearing in a component description must be those representing the inputs/outputs of the component in addition to assumables.

A ***structured system description*** is then a system structure and a set of component descriptions. In Figure 5 for example, $\Delta_A, \Delta_B, \ldots, \Delta_E$ are meant to be component descriptions.[4]

Our computational results in [3] can be summarized as follows. Given a structured system description with structure $\mathcal{G}$ and component descriptions $\Delta = \bigcup_i \Delta_i$, and given a system observation α, we can compute a negation normal form which is equivalent to the consequence $Cons^{\Delta}_{\mathbf{A}}(\alpha)$ in time and space that are exponential only in the width of the directed graph $\mathcal{G}$.

Moreover, if no assumables are shared between component descriptions, our results then guarantee that the computed negation normal form is also decomposable.[5]

The width of a graph w^* is a measure of the graph connectivity and is well known in the probabilistic and constraint-satisfaction communities since it underlies the computational complexity of some influential algorithms for belief and constraint networks [8, 7, 5, 10, 9, 11]. The less connected the graph is, the smaller that w^* is. In particular, if the system structure contains no undirected cycles, then w^* is the maximum number of parents per node, which corresponds to the maximum number of inputs per device component. In such a case, the size of a compiled system description is exponential only in the number of inputs that a component has (typically assumed to be a constant) and linear in all other aspects of the device.

It is important to note, however, that the width of a device structure may be small even if the device contains cycles. For example, the structure of an n-bit adder has a width of 4 and this width is independent of the number of bits n. Therefore, the size of a compiled n-bit adder grows linearly in n even though its structure is not cycle free. Table 1 depicts some concrete numbers showing the sizes of negation normal forms that result from compiling adders using a LISP implementation of our diagnostic compiler.

Table 1 implies that an on-line system for diagnosing a 6-bit adder needs to include only a propositional sentence in decomposable NNF which has 208 nodes and 446 arcs, in addition to a CSD evaluator.

Table 2 depicts a number of systems drawn from the benchmark circuits proposed in [2]. The number of components in each circuit is given, in addition to the size of the maximal clique obtained when compiling the device. The size of the maximal clique is an indication of the device structure width.[6] The compilation algorithm is exponential in the size of cliques and is linear in the number of such cliques. This is why we could not compile the first circuit in Table 2. The other three circuits, however, were compiled and the size of each resulting compilation is shown in the table.

[4]We view a device input as a component with one output and zero inputs.

[5]This condition can always be maintained but at the expense of adding auxiliary variables to the system structure. Further details can be found in [3].

[6]It is only an indication because of the following. Our algorithm converts a device structure into a jointree, a tree over sets-of-nodes known as cliques. There is more than one jointree corresponding to a particular structure, and each of these trees has a maximal clique size s_i. The structure width is $1 - \min_i s_i$. However, our algorithm is not guaranteed to generate the best jointree (that is, the one with the minimal s_i). Hence, the size of our maximal clique does not necessarily correspond to the structure width.

Table 1: Compiling n-bit adders.

Number of Bits	Number of Nodes	Number of Arcs
1	34	59
2	68	134
3	103	212
4	138	290
5	173	368
6	208	446

Table 2: Compiling digital circuits.

System	Number of Components	Max Clique	Number of Nodes (k)	Number of Arcs (k)
c432	196	22	–	–
c499	243	10	5	24
c880	443	10	11	28
c1355	587	10	10	37

6 Conclusion

We have described an approach for compiling devices which is marked by two characteristics: (1) the simplicity and linear complexity of its on-line phase; and (2) the computational complexity guarantees it offers on the off-line phase — the size of a compiled system description in particular. Although our compilation approach does not improve on the known computational complexity of diagnostic reasoning, it proposes a technique that has major implications on the practice of model-based diagnosis.

By separating diagnostic computations into off-line and on-line phases, one achieves three key objectives. First, most of the computational overhead is pushed into the off-line phase which needs to be performed only once, therefore, allowing for efficient on-line diagnostic systems. Second, this separation reduces the software/hardware demands of the on-line diagnostic system which needs to include only a CSD and its evaluator — as we have seen, a CSD has a relatively simple structure and its evaluator could be a very simple piece of software. Third, and most important practically, a CSD evaluator is so simple that it lends itself to cost-effective implementations on multiple software/hardware platforms. In our experience, this has been the most important practical aspect of our compilation approach since it allowed us to implement, quickly, real-world systems on various software/hardware platforms. We developed one diagnostic compiler on a LISP/Windows platform. But we could then deploy on-line diagnostic systems on any software/hardware platform for which a CSD evaluator can be made available.

What makes our compilation technique especially practical is the guarantees it offers on the size of a compiled system description. Such guarantees stand behind the success of many influential algorithms in the probabilistic and constraint-satisfaction literatures. We are unaware of any other model-based compilation approach for answering diagnostic queries which offers similar guarantees on the time to generate, and the space to store, a compiled system description.

Proofs

Proof of Theorem 1

Let β be a candidate diagnosis. We want to show that $\Delta \cup \{\alpha\} \not\models \neg\beta$ precisely when $CSD(\Delta, \mathbf{A}, \mathbf{O}) \wedge \alpha \not\models \neg\beta$. It suffices to show that $\Delta \cup \{\alpha\} \models \neg\beta$ precisely when $CSD(\Delta, \mathbf{A}, \mathbf{O}) \wedge \alpha \models \neg\beta$. It also suffices to show that $\Delta \models \neg\alpha \vee \neg\beta$ precisely when $CSD(\Delta, \mathbf{A}, \mathbf{O}) \models \neg\alpha \vee \neg\beta$. Note that $\neg\alpha \vee \neg\beta$ is an $(\mathbf{A} \cup \mathbf{O})$-sentence.

Suppose that $\Delta \models \neg\alpha \vee \neg\beta$. Then $CSD(\Delta, \mathbf{A}, \mathbf{O}) \models \neg\alpha \vee \neg\beta$ by Part 2 of Definition 1.

Suppose that $CSD(\Delta, \mathbf{A}, \mathbf{O}) \models \neg\alpha \vee \neg\beta$. Then $\Delta \models \neg\alpha \vee \neg\beta$ by Part 1 of Definition 1. ∎

Lemma 1 *Let γ_1 and γ_2 be $(\mathbf{A} \cup \mathbf{O})$-sentences and let α be an $\mathbf{O}$-instantiation where $\mathbf{A} \cap \mathbf{O}$ is empty. Then*

1. $(\gamma_1 \wedge \gamma_2) \mid \alpha = (\gamma_1 \mid \alpha) \wedge (\gamma_2 \mid \alpha)$.
2. $(\gamma_1 \vee \gamma_2) \mid \alpha = (\gamma_1 \mid \alpha) \vee (\gamma_2 \mid \alpha)$.
3. $(\neg\gamma_1) \mid \alpha = \neg(\gamma_1 \mid \alpha)$.
4. $\alpha \mid \alpha$ *is valid.* ∎

Proof of Theorem 2

1. This follows immediately since each **O**-literal in γ will be replaced by either *true* or *false* in $\gamma \mid \alpha$.

2. The proof is by induction on the structure of γ:

   (a) *Base case:* γ is a literal.
   If γ is an **A**-literal, then $\gamma \mid \alpha = \gamma$ and $\gamma \wedge \alpha \models \gamma$.
   If γ is an **O**-literal, then either
   - $\alpha \models \gamma$ and, hence, $\gamma \wedge \alpha \equiv \alpha$ and $\gamma \mid \alpha \equiv$ *true*; or
   - $\alpha \models \neg\gamma$ and, hence, $\gamma \wedge \alpha \equiv$ *false* and $\gamma \mid \alpha \equiv$ *false*.

   In either case, $\gamma \wedge \alpha \models \gamma \mid \alpha$.

   (b) *Inductive step:* $\gamma = \gamma_1 \wedge \gamma_2$.
   Assume that $\gamma_1 \wedge \alpha \models \gamma_1 \mid \alpha$ and $\gamma_2 \wedge \alpha \models \gamma_2 \mid \alpha$. Then $\gamma_1 \wedge \gamma_2 \wedge \alpha \models (\gamma_1 \mid \alpha) \wedge (\gamma_2 \mid \alpha)$ and $\gamma_1 \wedge \gamma_2 \wedge \alpha \models (\gamma_1 \wedge \gamma_2) \mid \alpha$ by Lemma 1. Therefore, $\gamma \wedge \alpha \models \gamma \mid \alpha$.

   (c) *Inductive step:* $\gamma = \gamma_1 \vee \gamma_2$.
   Same as Part (b).

3. We will first show that $\omega \models \gamma \mid \alpha$ only if there exists a world ω' which agrees with ω on **A** (written $\omega' \sim_{\mathbf{A}} \omega$) and $\omega' \models \gamma \wedge \alpha$.

   The proof is by induction on the structure of γ:

   (a) *Base case:* γ is a literal.
   If γ is an **A**-literal, then $\gamma \mid \alpha = \gamma$. Suppose $\omega \models \gamma \mid \alpha$. Then $\omega \models \gamma$. There is clearly a world $\omega' \sim_{\mathbf{A}} \omega$ such that $\omega \models \alpha$. Such a world must then satisfy $\omega' \models \gamma \wedge \alpha$.
   If γ is an **O**-literal, then either
   - $\alpha \models \gamma$ and, hence, $\gamma \wedge \alpha \equiv \alpha$ and $\gamma \mid \alpha \equiv$ *true*; or
   - $\alpha \models \neg\gamma$ and, hence, $\gamma \wedge \alpha \equiv$ *false* and $\gamma \mid \alpha \equiv$ *false*.

   In either case, the property holds.

   (b) *Inductive step:* $\gamma = \gamma_1 \wedge \gamma_2$.
   Assume the property holds for γ_1 and γ_2. Suppose $\omega \models \gamma \mid \alpha$. Then $\omega \models (\gamma_1 \wedge \gamma_2) \mid \alpha$, $\omega \models (\gamma_1 \mid \alpha) \wedge (\gamma_2 \mid \alpha)$, $\omega \models \gamma_1 \mid \alpha$ and $\omega \models \gamma_2 \mid \alpha$. By the induction hypothesis, there exists $\omega' \sim_{\mathbf{A}} \omega$ such that $\omega' \models \gamma_1 \wedge \alpha$ and there exists $\omega'' \sim_{\mathbf{A}} \omega$ such that $\omega'' \models \gamma_2 \wedge \alpha$. Note that ω' and ω'' must agree on **A** by definition and must also agree on **O** since they both satisfy α. Hence, there must exist a third world ω''' which agrees with ω' and ω'' on $\mathbf{A} \cup \mathbf{O}$. Moreover, for this world, $\omega''' \models \gamma_1 \wedge \gamma_2 \wedge \alpha$ and $\omega''' \models \gamma \wedge \alpha$.

   (c) *Inductive step:* $\gamma = \gamma_1 \vee \gamma_2$.
   Assume the property holds for γ_1 and γ_2. Suppose $\omega \models \gamma \mid \alpha$. Then $\omega \models (\gamma_1 \vee \gamma_2) \mid \alpha$, $\omega \models (\gamma_1 \mid \alpha) \vee (\gamma_2 \mid \alpha)$, $\omega \models \gamma_1 \mid \alpha$ or $\omega \models \gamma_2 \mid \alpha$. By the induction hypothesis, there exists $\omega' \sim_{\mathbf{A}} \omega$ such that $\omega' \models \gamma_1 \wedge \alpha$ or there exists $\omega' \sim_{\mathbf{A}} \omega$ such that $\omega' \models \gamma_2 \wedge \alpha$. Therefore, we have $\omega' \models (\gamma_1 \wedge \alpha) \vee (\gamma_2 \wedge \alpha)$, $\omega' \models (\gamma_1 \vee \gamma_2) \wedge \alpha$ and, hence, $\omega' \models \gamma \wedge \alpha$.

   Suppose now that β is an **A**-sentence and that $\gamma \wedge \alpha \models \beta$. We want to show that $\gamma \mid \alpha \models \beta$. It suffices to show that $\omega \models \gamma \mid \alpha$ only if $\omega \models \beta$.

   Suppose that $\omega \models \gamma \mid \alpha$. By the property we proved above, there exists a world $\omega' \sim_{\mathbf{A}} \omega$ such that $\omega' \models \gamma \wedge \alpha$. Moreover, $\omega' \models \beta$ by our supposition. Given that $\omega' \sim_{\mathbf{A}} \omega$ and since β is an **A**-sentence, we must have $\omega \models \beta$. ∎

Proof of Theorem 3

We want to show $CSD(\Delta, \mathbf{A}, \mathbf{O}) \mid \alpha \not\models \neg\beta$ precisely when $CSD(\Delta, \mathbf{A}, \mathbf{O}) \wedge \alpha \not\models \neg\beta$ for any candidate diagnosis β. It suffices to show that $CSD(\Delta, \mathbf{A}, \mathbf{O}) \mid \alpha \models \neg\beta$ precisely when $CSD(\Delta, \mathbf{A}, \mathbf{O}) \wedge \alpha \models \neg\beta$.

Suppose that $CSD(\Delta, \mathbf{A}, \mathbf{O}) \mid \alpha \models \neg\beta$. Then $CSD(\Delta, \mathbf{A}, \mathbf{O}) \wedge \alpha \models \neg\beta$ follows from Part 2 of Theorem 2.

Suppose that $CSD(\Delta, \mathbf{A}, \mathbf{O}) \wedge \alpha \models \neg\beta$. Then $CSD(\Delta, \mathbf{A}, \mathbf{O}) \mid \alpha \models \neg\beta$ from Part 3 of Theorem 2. ∎

References

[1] Jon Barwise, editor. *Handbook of Mathematical Logic.* North-Holland, Amsterdam, 1977.

[2] F. Beglez and H. Fujiwara. A neutral netlist of 10 combinational benchmark circuits and a target translator in fortran. In *Proceedings of the IEEE symposium on Circuits and Systems*, page http://www.cbl.ncsu.edu/www/CBL_Docs/iscas85.htn June, 1985.

[3] Adnan Darwiche. Model-based diagnosis using structured system descriptions. *Journal of Artificial Intelligence Research*, 1998. To appear.

[4] Johan de Kleer, Alan K. Mackworth, and Raymond Reiter. Characterizing diagnoses and systems. *Artificial Intelligence*, 56(2-3):197–222, 1992.

[5] Rina Dechter. Constraint networks. *Encyclopedia of Artificial Intelligence*, pages 276–285, 1992. S. Shapiro, editor.

[6] Kenneth D. Forbus and Johan de Kleer. *Building Problem Solvers.* MIT Press, 1993.

[7] Hector Geffner and Judea Pearl. An improved constraint-propagation algorithm for diagnosis. In *Proceedings of IJCAI*, pages 1105–1111, Milan, Italy, 1987.

[8] Cecil Huang and Adnan Darwiche. Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning*, 15(3):225–263, 1996.

[9] 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.

[10] A. K. Mackworth and E. C. Freuder. The complexity of some polynomial network consistency algorithms for constraint satisfaction problems. *Artificial Intelligence*, 25(1), 1985.

[11] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers, Inc., San Mateo, California, 1988.

[12] Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.

Explanatory Diagnosis: Conjecturing actions to explain observations

Sheila A. McIlraith*

Knowledge Systems Lab
Stanford University
Stanford, CA 94305-9020
sam@ksl.stanford.edu

Xerox PARC
3333 Coyote Hill Road
Palo Alto, CA 94301
mcilrait@parc.xerox.com

Abstract

In this paper we present contributions towards a logical theory of diagnosis for systems that can be affected by the actions of agents. Specifically, we examine the task of conjecturing diagnoses to explain *what happened* to a system, given a theory of system behaviour and some observed (aberrant) behaviour. We characterize what happened by introducing the notion of explanatory diagnosis in the language of the situation calculus. Explanatory diagnoses conjecture sequences of actions to account for a change in system behaviour. As such, we show that determining an explanatory diagnosis is analogous to classical AI planning with state constraints and incomplete knowledge. The representation scheme we employ provides an axiomatic solution to the frame, ramification and qualification problems. Exploiting this representation, we show that determining an explanatory diagnosis can be achieved by regression followed by theorem proving in the database describing what is known of the initial state of our system. Further, we show that by exploiting features inherent to diagnosis problems, we can simplify the diagnosis task.

1 INTRODUCTION

Given a theory of system behaviour and some observed aberrant behaviour, the traditional objective of diagnosis is to conjecture *what is wrong* with the system, (e.g., which components of the device are behaving abnormally, what diseases the patient is suffering from, etc.). Each candidate diagnosis consists of a subset of distinguished literals that are conjectured to be true or false in order to account for the observation in some way. Different criteria have been proposed for determining the space of such candidate diagnoses. Within formal accounts of diagnosis, two widely accepted definitions of diagnosis are consistency-based diagnosis (e.g., (Reiter 1987), (de Kleer et al. 1992)), and abductive explanation (e.g., (de Kleer et al. 1992), (Poole 1988), (Console and Torasso 1991), (McIlraith 1994)). Such research has historically focussed on static systems. Recently, some researchers have advocated extending diagnostic problem solving (DPS) to enable reasoning about actions, under the argument that DPS is purposive in nature and that systems operate within and are affected by agents[1].

In this paper we focus upon one aspect of diagnosing such dynamic systems. In particular, given a theory of system behaviour and some observation of (aberrant) behaviour, our concern is with the task of conjecturing diagnoses to explain *what happened* to the system (i.e., what actions or events occurred to result in the observed behaviour) (e.g., (Cordier and Thiébaux 1994), (McIlraith 1994a)). Knowing or conjecturing what happened is interesting in its own right, but it can also help to further constrain the space of possible states of the system. In so doing, conjecturing what happened assists in the prediction of what is wrong with a system, as well as predicting other relevant system behaviour. Compared to our traditional notion of *what is wrong* diagnoses, knowing *what happened* can more accurately capture the root cause of system malfunction rather than its manifestations, thus providing for the identification of future preventative as well as prescriptive actions.

In the spirit of previous foundational work in model-

* Much of this research was performed while the author was a doctoral candidate at the University of Toronto, Canada.

[1]An agent could be another system, a robot, a human, or nature.

based diagnosis (MBD) (e.g., (Reiter 1987), (Console and Torasso 1991), (de Kleer et al. 1992)), this paper presents a logical characterization of the diagnosis task. We take as our starting point the existing MBD research on characterizing diagnoses for static systems *without* a representation of actions (e.g., (de Kleer et al. 1992), (Console and Torasso 1991), (Reiter 1987)). Next, we exploit a situation calculus representation scheme previously proposed by the author (McIlraith 1997a) that enables the integration of a representation of action with the representation of the behaviour of a static system. With this representation in hand, we provide a logical characterization for the task of determining *what happened* to a system. The characterization is presented in the guise of *explanatory diagnosis*.

The distinguishing features of our characterization are afforded in great part by the richness of our representation scheme which provides a comprehensive and semantically justified representation of action and change. In particular, our representation provides an axiomatic closed-form solution to the frame and ramification problems, thus capturing the direct and indirect effects of actions in a compiled representation. This is critical to the ability to generate explanatory diagnoses efficiently. Further, our representation provides a closed-form solution to the qualification problem, thus identifying the conditions underwhich an action is possible. It is interesting to note that when we are dealing with incomplete knowledge of our initial state, conjecturing an action or sequence of actions also requires conjecturing that its preconditions are satisfied, which in many instances serves to further constrain our search.

As we show in the sections to follow, our characterization establishes a direct link between explanatory diagnosis and planning, deductive plan synthesis, and abductive planning. As a consequence of a completeness assumption embedded in our representation, we can exploit goal-directed reasoning in the form of regression (Waldinger 1977) in order to generate diagnoses. This completeness assumption also provides for an easy mapping of our situation calculus representation to Prolog. While explanatory diagnoses can be mathematically characterized in an analogous fashion to plans, an important distinction of explanatory diagnoses is that they can be refined to exploit features of our diagnosis problem that have no meaning in the context of planning. Indeed, we use diagnosis-specific attributes to define variants of explanatory diagnosis to deal with the challenges of incomplete knowledge and large search spaces.

2 REPRESENTATION SCHEME

2.1 SITUATION CALCULUS LANGUAGE

The situation calculus language in which we axiomatize our domains is a sorted first-order language with equality. The sorts are of type $\mathcal{A}$ for primitive *actions*, $\mathcal{S}$ for *situations*, and $\mathcal{D}$ for everything else, including domain objects (Lin and Reiter 1994). We represent each action as a (possibly parameterized) first-class object within the language. Situations are simply sequences of actions. The evolution of the world can be viewed as a tree rooted at the distinguished initial situation S_0. The branches of the tree are determined by the possible future situations that could arise from the realization of particular sequences of actions. As such, each situation along the tree is simply a history of the sequence of actions performed to reach it. The function symbol *do* maps an action term and a situation term into a new situation term. For example, $do(turn\_on\_pmp, S_0)$ is the situation resulting from performing the action of turning on the pump in situation S_0. The distinguished predicate $Poss(a, s)$ denotes that an action a is possible to perform in situation s (e.g., $Poss(turn\_on\_pmp, S_0)$). Thus, $Poss$ determines the subset of the situation tree consisting of situations that are possible in the world. Finally, those properties or relations whose truth value can change from situation to situation are referred to as *fluents*. For example, the fluent $on(Pmp, s)$ expresses that the pump is on in situation s.

The dialect of the situation calculus that we use in this paper is restricted to primitive, determinate actions. Our language does not include functional fluents, nor does it include a representation of time, concurrency, or complex actions, but we believe the results presented herein can be extended to more expressive dialects of the situation calculus (e.g., (Reiter 1996)).

2.2 DOMAIN REPRESENTATION

In this section we overview the representation scheme we use to characterize the system we will be diagnosing. The scheme, proposed in (McIlraith 1997a), integrates a situation calculus theory of action with a MBD system description, SD (de Kleer et al. 1992). The resulting representation of a system comprises both domain-independent and domain-specific axioms. The domain-independent axioms are the foundational axioms of the discrete situation calculus, Σ_{found} (Lin and Reiter 1994). They are analogous to the axioms of Peano arithmetic, modified to define the branching structure of our situation tree, rather than the number line. The domain-specific axioms, T specify both the *behaviour of the static system*, and the *actions*[2] that can affect the state of the system, as well as those actions required to achieve testing and repair. Together they define our situation calculus representation $\Sigma = \Sigma_{found} \wedge T$.

The domain-specific axioms composing T result from application of a procedure proposed in (McIlraith 1997a) that compiles a typical MBD system description, SD and a set of axioms relating to the preconditions and effects of actions into a representation that provides a closed-form solution to the frame, ramification and qualification problems. The resulting domain axiomatization $T = T_{SC}^{S_0} \wedge T_{domain} \wedge T_{SS} \wedge T_{AP} \wedge T_{UNA} \wedge T_{DCA} \wedge T_{S_0}$ is described below. The representation is predicated on an explicit causal ordering of fluents and an a completeness assumption. The assumption states that all the conditions underwhich an action a can lead, directly or indirectly, to fluent F becoming true or false in the successor state are captured in the axiomatization of our system.

We illustrate the representation scheme in terms of a small portion of a power plant feedwater system (McIlraith 1997) derived from the APACS project (Kramer et al. 1996). This simplified example models the filling of a vessel either by the operation of an electrically powered (Pwr) pump (Pmp), by manual filling, or by a siphon that was started by the pump or by manual filling. For notational convenience, all formulae are understood to be universally quantified with respect to their free variables, unless explicitly indicated otherwise. For a more thorough description of this representation scheme, and for a more extensive example please see ((McIlraith 1997a) and (McIlraith 1997)).

Every domain axiomatization, T comprises the following sets of axioms.

$$T_{SC}^{S_0} \wedge T_{domain} \wedge T_{SS} \wedge T_{AP} \wedge T_{UNA} \wedge T_{DCA} \wedge T_{S_0}$$

The set of state constraints relativized to situation S_0, $T_{SC}^{S_0}$ capture what is implicitly true about the initial database. They can be acquired from a typical MBD system description, SD as described in (McIlraith 1997a). In our simple example, $T_{SC}^{S_0}$ is as follows

$$\neg AB(Pwr, S_0) \wedge \neg AB(Pmp, S_0) \wedge on(Pmp, S_0) \supset filling(S_0) \quad (1)$$

$$manual\_fill(S_0) \supset filling(S_0) \quad (2)$$

$$\neg(on(Pmp, S_0) \wedge manual\_fill(S_0)) \quad (3)$$

The set of domain constraints, T_{domain} is as follows.

$$Pwr \neq Pmp \quad (4)$$

[2]Actions can be performed by agents: a human, another system, or nature.

The set of successor state axioms, T_{SS} is composed of axioms of the following general form, one for each fluent F.

$$Poss(a, s) \supset [F(do(a, s)) \equiv \Phi_F] \quad (5)$$

where Φ_F is a simple formula[3] of a particular syntactic form. Intuitively, a successor state axiom says the following:

$$\begin{aligned} Poss(a,s) \supset [fluent(do(a,s)) \equiv\ & \textit{an action made it true} \\ \vee\ & \textit{a state constraint made it true} \\ \vee\ & \textit{it was already true} \\ & \wedge\ \textit{neither an action nor a state constraint made it false}]. \end{aligned}$$

The following axioms compose T_{SS} for our example.

$$\begin{aligned} Poss(a,s) \supset [on(Pmp, do(a,s)) \equiv a = turn\_on\_pmp \\ \vee (on(Pmp,s) \wedge a \neq turn\_off\_pmp)] \end{aligned} \quad (6)$$

$$\begin{aligned} Poss(a,s) \supset [AB(Pwr, do(a,s)) \equiv a = pwr\_failure \\ \vee (AB(Pwr,s) \wedge a \neq aux\_pwr \wedge a \neq pwr\_fix)] \end{aligned} \quad (7)$$

$$\begin{aligned} Poss(a,s) \supset [AB(Pmp, do(a,s)) \equiv \\ a = pmp\_burn\_out \\ \vee (AB(Pmp,s) \wedge a \neq pmp\_fix)] \end{aligned} \quad (8)$$

$$\begin{aligned} Poss(a,s) \supset [manual\_fill(do(a,s)) \equiv \\ a = turn\_on\_manual\_fill \\ \vee (manual\_fill(s) \\ \wedge a \neq turn\_off\_manual\_fill)] \end{aligned} \quad (9)$$

$$\begin{aligned} Poss(a,s) \supset [& filling(do(a,s)) \equiv \\ & a = turn\_on\_manual\_fill \\ & \vee (manual\_fill(s) \wedge a \neq turn\_off\_manual\_fill) \\ & \vee [(a \neq pwr\_failure \\ & \quad \wedge (\neg AB(Pwr,s) \vee a = aux\_pwr \\ & \quad \quad \vee a = pwr\_fix)) \\ & \wedge (a \neq pmp\_burn\_out \\ & \quad \wedge (\neg AB(Pmp,s) \vee a = pmp\_fix)) \\ & \wedge (a = turn\_on\_pmp \\ & \quad \vee (on(Pmp,s) \wedge a \neq turn\_off\_pmp))] \\ & \vee (filling(s) \wedge a \neq stop\_siphon)] \end{aligned} \quad (10)$$

Axiom (6) states that if action a is possible in situation s, then the pump is on in the situation

[3]A *simple formula* only mentions domain-specific predicate symbols, fluents do not include the function symbol *do*, there is no quantification over sort *situation*, and there is at most one free *situation* variable.

resulting from performing action a in situation s (i.e., $on(Pmp, do(a,s))$) if and only if the action a is $turn\_on\_pmp$, or the pump was already on in s and a was not the action $turn\_off\_pmp$.

The set of action precondition axioms, T_{AP} is composed of axioms of the following general form, one for each action prototype A in the domain.

$$Poss(A(\vec{x}), s) \equiv \Pi_A \tag{11}$$

where Π_A is a simple formula with respect to s, capturing the necessary and sufficient conditions for prototype action $A(\vec{x})$ to be executable in situation s.

$$Poss(stop\_siphon, s) \equiv (\neg manual\_fill(s) \wedge \neg on(Pmp, s)) \tag{12}$$

$$Poss(pmp\_fix, s) \equiv \neg on(Pmp, s) \tag{13}$$

$$Poss(pmp\_burn\_out, s) \equiv on(Pmp, s) \tag{14}$$

$$Poss(turn\_on\_manual\_fill, s) \equiv \neg on(Pmp, s) \tag{15}$$

$$Poss(turn\_on\_pmp, s) \equiv \neg manual\_fill(s) \tag{16}$$

$$Poss(turn\_off\_pmp, s) \equiv on(Pmp, s) \tag{17}$$

$$Poss(turn\_off\_manual\_fill, s) \equiv manual\_fill(s) \tag{18}$$

$$Poss(pwr\_failure, s) \equiv \tag{19}$$

$$Poss(pwr\_fix, s) \equiv Poss(aux\_pwr, s) \equiv \text{true} \tag{20}$$

Finally, we provide a possible set of initial conditions for our system, T_{S_0}. These constitute the explicit aspect of the initial database. Note that in general we do not have complete knowledge of the initial state of our system. This makes the task of diagnosis all the more challenging. In this example, we do not know initially whether the pump and power are operating normally. We also do not know whether the vessel was filling in the initial state.

$$on(Pmp, S_0) \wedge \neg manual\_fill(S_0) \tag{21}$$

T_{UNA} and T_{DCA} are the unique names axioms for actions and the domain closure axiom for actions, respectively.

This concludes the description of our representation scheme. Before advancing to issues of diagnosis, we note (Reiter 1998) that our proposed situation calculus representation can be viewed as an executable specification because it is easily realized in Prolog by exploiting Prolog's completion semantics and simply replacing the equivalence connectives characteristic of axioms in T_{SS} and T_{AP} by implication connectives. The Lloyd-Topor transformation (Lloyd 1987) must then be applied to convert this theory into Prolog clausal form. Later in this paper, we will advocate using Waldinger's notion of regression to rewrite axioms of our representation and simplify computation. This type of regression rewriting is precisely achieved by Prolog's backwards chaining mechanism.

3 PRELIMINARIES

With our representation in hand, we turn our attention to the task of diagnosis. In this section we introduce the framework for performing diagnosis relative to our representation. For our purposes we adopt the ontological and notational convention of the MBD literature and view the systems we are diagnosing as comprising a number of interacting *components*, $COMPS$. These components have the property of being either abnormal or normal in a situation. We express this property in our situation calculus language using the fluent AB. For example, $AB(Pmp, s)$ denotes that the pump component is abnormal in situation s. Note that the use of AB is not mandatory to the contributions of this paper. Once again, following the convention in the MBD literature, we define our diagnoses relative to the domain-independent concept of a *system* (de Kleer et al. 1992), adapted to our situation calculus framework.

Definition 1 (System)
A system is a quadruple $(\Sigma, HIST, COMPS, OBS)$ *where:*

- Σ*, the background theory, is a set of situation calculus sentences describing the behaviour of our system and the actions that can affect it.*
- $HIST$*, the history, is a sequence of ground actions* $[a_1, \ldots, a_k]$ *that were performed starting in* S_0.
- $COMPS$*, the components, is a finite set of constants.*
- OBS_F*, the observation, is a simple formula composed of fluents whose only free variable is the situation variable s, and which are otherwise ground.*

Example 1
In our power plant example above, Σ *is our axiomatization* $\Sigma_{found} \wedge T$ *and* $COMPS = \{Pmp, Pwr\}$*. The observation,* OBS_F *could be* $filling(s)$*, for example.* $HIST$ *could be empty, i.e.,* $[\ ]$*, or perhaps* $[turn\_on\_pmp]$.

4 EXPLANATORY DIAGNOSIS

In this section we introduce and formally characterize the notion of an explanatory diagnosis which conjectures *what happened* to result in some observed (aberrant) behaviour. Given a system, $(\Sigma, HIST, COMPS, OBS_F)$, the objective of explanatory diagnosis is to conjecture a sequence of actions, $[\alpha_1, \ldots, \alpha_n]$

such that our observation is true in the situation resulting from performing that sequence of actions in $do(HIST, S_0)$.

Definition 2 (Explanatory Diagnosis)
An explanatory diagnosis for system $(\Sigma,\ HIST,\ COMPS,\ OBS_F)$ *is a sequence of actions* $E = [\alpha_1, \ldots, \alpha_n]$ *such that,*

- $\Sigma \models Poss(HIST \cdot E, S_0)^3 \wedge OBS_F(do(HIST \cdot E, S_0))$.

Thus, E is an explanatory diagnosis when the observation is true in the situation resulting from performing the sequence of actions E in situation $do(HIST, S_0)$, and further that the preconditions for each action of the action sequence $HIST \cdot E$ are true in the appropriate situations, commencing at S_0.

The problem of determining explanatory diagnoses is an instance of temporal explanation or postdiction (e.g., (Shanahan 1993)), and is related to the classical AI planning problem. In particular, identifying the sequence of actions composing an explanatory diagnosis, E is analogous to the plan synthesis problem, and thus is realizable using deduction on the situation calculus axioms. According to (Green 1969), a plan to achieve a goal $G(s)$ is obtained as a side effect of proving $Axioms \models \exists s.G(s)$. The binding for the situation variable s represent the sequence of actions. In our case, $Axioms \models \exists s.G(s)$ is analogous to $\Sigma \models \exists s.OBS_F(s)$. As such, our representation enables us to generate explanatory diagnoses deductively, just as we could deductively generate a plan in the situation calculus. Note that the task of generating explanatory diagnoses is analogous to plan synthesis in the presence of state constraints – a challenging problem. Our representation scheme eliminates the additional challenges presented by state constraints by providing a domain axiomatization that a priori solves the frame, ramification and qualification problems.

Example 2
Continuing with our power plant example, given the system $(\Sigma, [\,], \{Pwr, Pmp\}, \neg filling(s))$, *the sequence of actions* $[pwr\_failure]$ *constitutes one example of an explanatory diagnoses for the system. Another explanatory diagnosis for our system is* $[turn\_off\_pmp]$.

^{3}**Notation:**
$HIST \cdot E$ is an abbreviation for $[a_1, \ldots, a_k, \alpha_1, \ldots, \alpha_n]$.
$do([a_1, \ldots, a_m], s)$ is an abbreviation for
$do(a_m,(do(a_{m-1},(do(a_{m-2},(\ldots,(do(a_1,s))))))))$.
Finally, $Poss([a_1, \ldots, a_n], s)$ is an abbreviation for $Poss(a_1, s) \wedge Poss(a_2, do(a_1, s)) \wedge \ldots \wedge Poss(a_n, do([a_1, \ldots, a_{n-1}], s))$.

Observe that for certain problems there can be an infinite number of sequences of actions that constitute explanatory diagnoses. For example, the following sequences of actions also constitute valid explanatory diagnoses for our example system:

$[pwr\_failure, pwr\_fix, pwr\_failure]$,
$[pwr\_failure, aux\_pwr, pwr\_failure]$,
$[turn\_off\_pmp, pwr\_failure, turn\_on\_pmp]$,
and so on.

Definition 2 is not sufficiently discriminating to eliminate these, clearly suboptimal explanatory diagnoses. We must define a preference criterion. Probability measures, even simple order of magnitude probabilities have provided an effective preference criterion for many applications of MBD (de Kleer 1991). Likewise, in the case of determining explanatory diagnoses in the context of the situation calculus, probabilities will serve us well in identifying preferred explanatory diagnoses. Unfortunately, probability measures are not always available and the correct treatment of probabilities in our situation calculus framework is only now being developed. In this paper, we limit our discussion to what we refer to as a chronologically simple preference criterion.

In our chronologically simple preference criterion, we prefer diagnoses that are relativized to situations reached without performing any extraneous actions. Note that this preference criterion is syntactic in nature, relying on the notion of a primitive action as a unit measure.

Definition 3 (Simpler)
Given a sequence of actions $HIST = [\alpha_1, \ldots, \alpha_n]$, *define* $ACTS(HIST)$ *to be the set* $\{\alpha_1, \ldots, \alpha_n\}$, *and* $LEN(HIST)$ *to be the the length of the sequence of actions composing* $HIST$.

Thus, given $HISTA = [a_1, \ldots, a_n]$ *and* $HISTB = [b_1, \ldots, b_n]$, *situation* $S_A = do(HISTA, S_0)$ *is simpler than situation* $S_B = do(HISTB, S_0)$ *iff* $ACTS(HISTA) \subseteq ACTS(HISTB)$ *and* $LEN(HISTA) < LEN(HISTB)$.

Definition 4 (Chronologically Simple Diag.)
E is a chronologically simple explanatory diagnosis for system $(\Sigma,\ HIST,\ COMPS,\ OBS_F)$ *iff E is an explanatory diagnosis for the system, and there is no explanatory diagnosis* E' *such that situation* $S' = do(HIST \cdot E', S_0)$ *is simpler than situation* $S = do(HIST \cdot E, S_0)$.

We might further distinguish this criterion to prefer chronologically simple explanatory diagnoses comprised solely of actions performed by nature.

Finally, observe that the characterization of explanatory diagnosis just presented assumes that E and OBS_F occur *after* $HIST$. While this assumption is not critical to characterizing explanatory diagnoses, it acts as a form of preference, facilitating computation of E.

5 EXPLOITING REGRESSION

In the previous section, we provided a characterization of explanatory diagnosis. We observed that computing an explanatory diagnosis for system (Σ, $HIST$, $COMPS$, OBS_F) is analogous to generating a plan to achieve a goal $OBS_F(s)$ starting with axioms Σ and situation $do(HIST, S_0)$. At first glance, the general problem of computing explanatory diagnoses does not look very promising for at least three reasons: the second-order induction axiom in Σ_{found}, the potential incompleteness of the initial database, and the potentially large size of the situation search space. In this section, we show how diagnoses can be computed by exploiting regression (Waldinger 1977). We are able to exploit regression as a direct result of the embedded completeness assumption inherent in our representation scheme. In this context, regression is a recursive rewriting procedure that we use to reduce the nesting of the *do* function in situation terms, or to eliminate the *Poss* predicate. We show that generating explanatory diagnoses reduces to regression followed by entailment with respect to the initial database. Computationally, the merit of regression is that it searches backwards through the situation space from the observation rather than searching forward from the initial database. Under the assumption that the observation consists of fewer literals than the initial database, regression will make for more efficient search. Observe that Prolog's backwards chaining mechanism achieves the substitution performed by regression.

Following directly in the spirit of previous work by Reiter ((Reiter 1991), (Reiter 1992)) and more recently (Reiter 1998) on the exploitation of regression for planning and query answering, we first define two regression operators, $\mathcal{R}^*$ and $\mathcal{R}_{Poss}$.

Definition 5 (Regression Operator $\mathcal{R}^*$) *Given a set of successor state axioms, T_{SS} composed of axioms of the form $Poss(a,s) \supset [F(do(a,s)) \equiv \Phi_F]$, $\mathcal{R}^*[\Psi]$, the repeated regression of formula Ψ with respect to successor state axioms T_{SS} is the formula that is obtained from Ψ by repeatedly replacing each fluent $F(do(a,s))$ in Ψ by Φ_F, until the resulting formula makes no mention of the function symbol do.*

For example,

$$\begin{aligned}
&\mathcal{R}^*[on(Pmp, do(\mathit{turn\_on\_manual\_fill}, do(\mathit{turn\_on\_pmp}, S_0)))] \\
&= \mathcal{R}^*[(\mathit{turn\_on\_manual\_fill} = \mathit{turn\_on\_pmp}) \\
&\qquad \vee\ (on(Pmp, do(\mathit{turn\_on\_pmp}, S_0)) \\
&\qquad\qquad \wedge\ (\mathit{turn\_on\_manual\_fill} \neq \mathit{turn\_off\_pmp}))] \\
&= \mathcal{R}^*[\text{false} \vee (on, (Pmp, do(\mathit{turn\_on\_pmp}, S_0)) \wedge \text{true})] \\
&= \mathcal{R}^*[(\mathit{turn\_on\_pmp} = \mathit{turn\_on\_pmp}) \\
&\qquad \vee\ (on(pmp, S_0)) \wedge (\mathit{turn\_on\_pmp} \neq \mathit{turn\_off\_pmp}))] \\
&= \text{true}
\end{aligned}$$

We can similarly define a $Poss$ regression operator over the set of action precondition axioms, T_{AP}. This regression operation rewrites each occurrence of the literal $Poss(a,s)$ by Π_A as defined in the action precondition axioms.

Definition 6 (Regression Operator $\mathcal{R}_{Poss}$) *Given a set of action precondition axioms, T_{AP} composed of axioms of the form $Poss(A(\vec{x}), s) \equiv \Pi_A$, $\mathcal{R}_{Poss}[W]$ is the formula obtained by replacing each occurrence of predicate $Poss(A(\vec{x}), s)$ by Π_A. All other literals of W remain the same.*

Reiter proved soundness and completeness results for regression applied to a theory with no state constraints (Theorem 1, Theorem 2, (Reiter 1992)). In the theorem below, we prove soundness and completeness results for our representation scheme which includes state constraints. The theory Σ_{init} mentioned in the theorem below is a subset of Σ containing only information about the initial situation, and no information about successor situations. It also excludes the induction axiom of Σ_{found}.

Theorem 1 (Soundness & Completeness)
Given

- *Σ_{init}, a subset of the situation calculus theory Σ, such that $\Sigma_{init} = \Sigma_{UNS} \wedge T_{S_0} \wedge T_{SC}^{S_0} \wedge T_{domain} \wedge T_{UNA}$,*

  where Σ_{UNS} is a subset of Σ_{found} containing the set of unique names axioms for situations.

- *a sequence of ground actions, s_HIST such that $\Sigma_{init} \wedge \mathcal{R}^*\ [\mathcal{R}_{Poss}\ [Poss(s\_HIST,\ S_0)]]$ is satisfiable.*

- *$Q(s)$, a simple formula whose only free variable is the situation variable s.*

Suppose $S = do(s\_HIST, S_0)$, then

- *$\Sigma \models Q(do(s\_HIST, S_0))$ iff*
  $\Sigma_{init} \models \mathcal{R}^*[Q(do(s\_HIST, S_0))]$,

- *$\Sigma \models Poss(s\_HIST, S_0)$ iff*
  $\Sigma_{init} \models \mathcal{R}^*[\mathcal{R}_{Poss}[Poss(s\_HIST, S_0)]]$,

- *$\Sigma \wedge Poss(s\_HIST, S_0) \wedge Q(do(s\_HIST, S_0))$ is satisfiable iff $\Sigma_{init} \wedge \mathcal{R}^*[\mathcal{R}_{Poss}[Poss(s\_HIST, S_0)]] \wedge \mathcal{R}^*[Q(do(s\_HIST, S_0))]$ is satisfiable.*

Thus, assuming situation s is a possible situation and exploiting regression, $Q(s)$ holds at situation s iff its regression is entailed in the initial database. The beauty of Theorem 1 is that it enables us to generate explanatory diagnoses via regression followed by theorem proving in the initial database, without the need for the second-order induction axiom in Σ_{found}. ¿From these results, we can characterize explanatory diagnosis with respect to regression.

Corollary 1 (Expl. Diagnosis w/ Regression) *The sequence of actions $E = [\alpha_1, \ldots, \alpha_k]$ is an explanatory diagnosis for system (Σ, $HIST$, $COMPS$, OBS_F) iff*

- $\Sigma_{init} \models \mathcal{R}^*[\mathcal{R}_{Poss}[Poss(HIST \cdot E, S_0)]] \wedge \mathcal{R}^*[OBS_F(do(HIST \cdot E, S_0))]$.

6 EXPLOITING THE TASK

In the previous section we showed that we could exploit regression to simplify the computation of explanatory diagnoses. A remaining source of difficulty in generating explanatory diagnoses is that our search space may be large and our initial database may be incomplete. An incomplete initial database both underconstrains our search problem, and precludes us from using certain planning machinery, such as STRIPS (Fikes and Nilsson 1971), that assumes a complete or near-complete initial database (Lin and Reiter 1995). In this section we show how to exploit features of diagnosis problems to further assist in the generation of explanatory diagnoses. In particular, we propose to 1) make assumptions regarding our domain, 2) relax our criteria for explanatory diagnoses, and 3) verify rather than generate diagnoses. An additional means of simplifying our computational task is to use likelihoods of actions and action sequences to focus search for explanatory diagnoses. Detailed discussion of this option is beyond the scope of this paper.

6.1 ASSUMPTION-BASED DIAGNOSES

In diagnostic problem solving it is common to make further assumptions that are consistent with what we know of the world. For example, we may assume that in the absence of information to the contrary, components are operating normally, or certain properties hold of the world. To support such assumption-based reasoning, we define the notion of an assumption-based explanatory diagnosis.

Definition 7 (Assumption-based Expl. Diag.) *Given an assumption $H(S)$ relativized to ground situation S such that*

- $S_0 \leq^4 S \leq do(HIST \cdot E, S_0)$,
- *$\Sigma \wedge H(S)$ is satisfiable, and*
- $\Sigma \wedge H(S) \models Poss(HIST, S_0)$.

An assumption-based explanatory diagnosis for system (Σ, $HIST$, $COMPS$, OBS_F) under assumption $H(S)$ is a sequence of actions $E = [\alpha_1, \ldots, \alpha_k]$ such that,

- $\Sigma \wedge H(S) \models Poss(HIST \cdot E, S_0) \wedge OBS_F(do(HIST \cdot E, S_0))$.

In Example 2 of the previous section, we did not have complete information about the initial state of our system. It could actually have been the case that observation $\neg filling$ was true in S_0, i.e., $\neg filling(S_0)$, but since it was not entailed by Σ, the empty action sequence was not proposed as a valid explanatory diagnosis, and we were forced to conjecture a sequence of actions to account for our observation. If we assume $\neg filling(S_0)$, then the empty sequence of actions is indeed an assumption-based explanatory diagnosis.

In generating explanatory diagnoses, we may want to make a priori assumptions about the world, conjoin these assumptions to our theory and then try to compute our explanatory diagnoses. For example, we may wish to assume that all components are operating normally in S_0. This would be achieved by making $H(S)$ in our definition above equal to $\bigwedge_{c \in COMPS} \neg AB(c, S_0)$ (i.e., $\neg AB(Pmp, S_0) \wedge \neg AB(Pwr, S_0)$). Similarly, we may wish to assume that the observation, OBS_F is true in $do(HIST, S_0)$. In our example above, this would mean assuming $\neg filling(S_0)$. In other instances, we might want $H(S)$ to equal a *what is wrong* diagnosis that we are currently entertaining, relativized to a previous situation, e.g., $AB(Pmp, S_0)$.

In still other instances, we may not want to fix our assumptions a priori but rather make the minimum number of assumptions necessary to generate a chronologically simple explanatory diagnosis. Such assumptions might be limited to a distinguished set of literals which the domain axiomatizer considers to be legitimately assumable (e.g., AB fluents). There might also be a partial ordering on such assumables fluents.

[4]**Notation:** The transitive binary relation $<$ defined in Σ_{found} further limits our situation tree by restricting the actions that are applied to a situation to those whose preconditions are satisfied in the situation. Intuitively, if $s < s'$, then s and s' are on the same branch of the tree with s closer to S_0 than s'. Further, s' can be obtained from s by applying a sequence of actions whose preconditions are satisfied by the truth of the $Poss$ predicate.

The distinction regarding when we make our assumptions affects the machinery by which we compute assumption-based explanatory diagnoses. If assumptions are made prior to computation we simply conjoin the regression of the assumption to the initial database and use regression and theorem proving as we would for generating normal explanatory diagnoses. When assumptions are interleaved with computation, generating an assumption-based explanatory diagnosis requires abduction.

In a theorem prover, abduction is generally implemented as proof-tree completion, i.e., by resolving dead-ends of proof trees with abducible literals. To generate an explanatory diagnosis, an abductive theorem prover would attempt to prove $OBS_F(s)$. If an attempted proof failed because it dead-ended on a literal or literals that were assumable, then these would be abduced and the proof continued. We have not yet implemented such an abduction engine for our situation calculus system. It is interesting to note that in the context of planning, (Shanahan 1997) has used abduction to implement a partial order planner for the event calculus. In contrast to the deductive approach of the situation calculus, this planner abduces actions and the relative order of certain actions.

6.2 POTENTIAL DIAGNOSES

In addition to making assumptions to help complete our theory, we can also facilitate computation by relaxing the criteria for defining an explanatory diagnosis. To this end, we observe that the requirement in Definitions 2 and 7 that $\Sigma \models Poss(HIST \cdot E, S_0)$ may be too stringent in the case of an incomplete initial database. That is, it may not be reasonable to require that we know that an action is possible in a situation that is incompletely specified. We may prefer to consider explanatory diagnoses, where the theory allows us to consistently assume that the preconditions for $HIST$ or for $HIST \cdot E$ hold, but not necessarily that they are entailed by our theory. To this end, we propose the following alteration to our definition of explanatory diagnoses, Definition 2. A comparable refinement can be made to our definition of assumption-based explanatory diagnosis, Definition 7.

Definition 8 (Potential Explanatory Diagnosis) *A potential explanatory diagnosis for system* (Σ, $HIST$, $COMPS$, OBS_F) *is a sequence of actions* $E = [\alpha_1, \ldots, \alpha_k]$ *such that,*

- $\Sigma \wedge Poss(HIST \cdot E, S_0)$ *is satisfiable, and*
- $\Sigma \wedge Poss(HIST \cdot E, S_0) \models$
  $OBS_F(do(HIST \cdot E, S_0))$.

Note that $HIST$ is a sequence of actions that we know to have been performed.Thus, the preconditions for each of the actions in $HIST$ are true in the corresponding situations. This provides us with further information concerning the truth values of fluents at various situations, helping to constrain our search.

6.3 VERIFYING LIKELY DIAGNOSES

To further address the problem of generating explanatory diagnoses, we propose exploiting domain information and maintaining a library of (assumption-based) explanatory diagnoses, indexed by observations and/or situation histories. With candidate diagnoses in hand, the problem of computing explanatory diagnoses reduces to a verification problem, rather than a generation problem. Given a system (Σ, $HIST$, $COMPS$, OBS_F), and a candidate diagnosis E, such that $S = do(HIST \cdot E, S_0)$, we are interested in verifying that E is indeed a diagnosis of the system. Verifying a candidate diagnosis is simply a query evaluation problem. It can be accomplished by regression and theorem proving in the initial database, as per Theorem 1 above.

Example 3
Given the system (Σ, [ ], $\{Pwr, Pmp\}$, $\neg filling(s)$), *and the candidate diagnosis* E=[$pwr\_failure$], E *can be verified to be an explanatory diagnosis with respect to the system by evaluating the query*

$$\mathcal{R}^*[\mathcal{R}_{Poss}[Poss(do(pwr\_failure, S_0))]] \\ \wedge \mathcal{R}^*[\neg filling(do(pwr\_failure, S_0))]$$

with respect to the initial database, Σ_{init}.

7 RELATED WORK

This work has been influenced by formal characterizations of diagnosis for systems without an explicit representation of actions (e.g., (de Kleer et al. 1992), (Reiter 1987), (Console and Torasso 1991)) and by Reiter's work on the frame problem and the problem of temporal projection (Reiter 1992) and (Reiter 1998). Aside from previous work by the author (e.g., (McIlraith 1997), (McIlraith 1997a)), research to date has not explicitly addressed the problem of integrating a rich representation of actions into diagnostic reasoning. As such, there is little related work that exploits a comprehensive representation of action.

A recent notable exception is Thielscher's work on dynamic diagnosis (Thielscher 1997). Thielscher's basic representation is similar in spirit to ours, though he does not use the situation calculus and he does not exploit an axiomatic solution to the frame, ramification and qualification problems. Like us, he adopts the basic MBD ontology, employs an action theory to represent actions, and represents certain state constraints causally to capture the indirect effects of actions as dictated by the system description of the device. Where he differs is in the actual task he is performing, and the assumptions he makes. To use terminology discussed here, he is computing *what is wrong* diagnoses. Given an action history and some observed aberrant behaviour, Thielscher conjectures that certain components must be abnormal to account for the observed behavior. He is not conjecturing actions. To simplify computation, he assumes that all components are initially normal, and he uses a priori likelihood of failure to select most likely candidate diagnoses. While his paper examines *what is wrong* diagnoses, rather than *what happened* diagnoses, clearly the two are intimately related. (McIlraith 1997) discusses the interrelationship between these two types of diagnosis in the context of the situation calculus framework described here.

The research on temporal diagnosis and diagnosis of dynamic systems originating in the diagnosis research community (e.g., (Brusconi et al. 1995), (Console et al. 1994), (Hamscher 1991), (Friedrich and Lackinger 1991), (Lackinger and Nejdl 1991), (Dressler 1994)) and in particular (Cordier and Thiébaux 1994) is also loosely related. (Brusconi et al. 1995) recently provided a characterization of temporal abductive diagnosis together with algorithms for computing these diagnoses under certain restrictions. Building on earlier work (Console et al. 1994), they decouple atemporal and temporal diagnoses, using SD to represent the behaviour of the atemporal components and transition graphs to represent the temporal components. The later work uses temporal constraints to represent the temporal components. Also related is the work on event-based diagnosis by (Cordier and Thiébaux 1994). Their work is similar in motivation to our work on explanatory diagnosis, viewing the diagnosis task as the determination of the event-history of a system between successive observations. While this work is related, the representation of action is impoverished. It does not provide a comprehensive representation of the preconditions for and the effects of actions, nor does it address the frame, ramification and qualification problems. In particular, their notion of actions is simply an explicit transition systems. This is sufficiently expressive for their application but the lack of a compact representation proves problematic.

In the area of reasoning about action, research on temporal explanation and postdiction has an interesting relationship to this work (e.g., (Crawford and Etherington 1992), (Baker 1991)). Of particular note is Shanahan's research (Shanahan 1993). While Shanahan also proposes the situation calculus as a representation language for axiomatizing his domain, he does so without an axiomatic solution to the frame and ramification problems. As such these problems must be addressed coincidentally with generating explanatory diagnoses. In contrast, our characterization of explanatory diagnosis, with its axiomatic solution to the frame and ramification problems, enables simpler characterization and computation of temporal explanation.

8 SUMMARY

The results in this paper provide contributions to model-based diagnosis and knowledge representation. Our concern in this paper was, given a system that affects and can be affected by the actions of agents, and given some observed (aberrant) behaviour, how do we capture the notion of *what happened*, i.e., how do we go about conjecturing a sequence of actions that account for the behaviour we have observed.

We addressed this problem by providing a mathematical characterization of the notion of explanatory diagnosis in the context of a rich situation calculus representation, proposed in (McIlraith 1997a). Our characterization made apparent the direct relationship of explanatory diagnosis to the planning task, and in particular to Green's notion of deductive plan synthesis. However, generating explanatory diagnoses is actually akin to planning in the face of state constraints and a potentially incomplete initial database. Our representation scheme addressed the challenges presented by state constraints by providing an axiomatic a priori solution to the frame, ramification and qualification problems. This enabled us to extend Reiter's results on the soundness and completeness of regression and to show that we can generate explanatory diagnoses by regression followed by theorem proving in the initial database. A remaining difficulty was that our initial database is often incomplete. Exploiting features of diagnosis problems, we proposed the notions of assumption-based and potential explanatory diagnosis, to allow for the conjectured sequences of actions that constitute a diagnosis to be predicated on some other assumptions we choose to make about the world. Finally, we proposed exploiting a library of precomputed likely diagnoses, indexed by context and observations. This enabled us to verify, rather than generate explanatory diagnoses.

In our dissertation work (McIlraith 1997), we have also addressed the complementary problem of conjecturing *what is wrong* diagnoses. In future work we examine the application of GOLOG procedures (Levesque et al. 1997) to diagnostic problem solving.

Acknowledgements

I would like to thank Ray Reiter, Yves Lespérance, Hector Levesque and Allan Jepson for helpful comments on the work presented in this paper. I would also like to thank Oskar Dressler, for engaging me in an interesting discussion regarding this work that helped me look at the material in a different light. Finally, thanks to the reviewers for their thoughtful reviews.

References

A. Baker (1991). Nonmonotonic Reasoning in the Framework of the Situation Calculus. *Artificial Intelligence* **49**(5–23):361–392.

V. Brusconi, L. Console, P. Terenziani and D. Theseider Dupré (1995). An Efficient Algorithm for Computing Temporal Abductive Diagnoses. In *Proceedings of the Sixth International Workshop on Principles of Diagnosis*, 41–48.

L. Console, L. Portinale, D. Theseider Dupré and P. Torasso (1994). Diagnosing Time-Varying Misbehavior: An Approach Based on Model Decomposition. *Annals of Mathematics and Artificial Intelligence* **11**(1–4):381–398.

L. Console and P. Torasso (1991). A Spectrum of Logical Definitions of Model-Based Diagnosis. *Computational Intelligence* **7**(3):133–141.

M. Cordier and S. Thiébaux (1994). Event-Based Diagnosis for Evolutive Systems. In *Proceedings of the Fifth International Workshop on Principles of Diagnosis*, 64–69.

J. Crawford and D. Etherington (1992). Formalizing Reasoning About Change: A Qualitative Reasoning Approach. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, 577–582.

J. de Kleer (1991). Focusing on Probable Diagnoses. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, 842–848.

J. de Kleer, A.K. Mackworth and R. Reiter (1992). Characterizing Diagnoses and Systems. *Artificial Intelligence* **56**(2–3):197–222.

O. Dressler (1994). Model-based Diagnosis on Board: Magellan-MT Inside. In *Proceedings of the Fifth International Workshop on Principles of Diagnosis*, 87–92.

R. Fikes and N. Nilsson (1971). STRIPS: A New Approach to Theorem Proving in Problem Solving. *Artificial Intelligence* **2**:189–208.

G. Friedrich and F. Lackinger (1991). Diagnosing Temporal Misbehaviour. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, 1116–1122.

C. Green (1969). Theorem Proving by Resolution as a Basis for Question-Answering Systems. In B. Meltzer and D. Michie (ed.), *Machine Intelligence 4*, 183–205. New York: American Elsevier.

W. Hamscher (1991). Modeling Digital Circuits for Troubleshooting. *Artificial Intelligence* **51**(1–3):223–271.

B. Kramer et al. (1996). Developing an Expert System Technology for Industrial Process Control: An Experience Report. In *Proceedings of the Conference of the Canadian Society for Computational Studies of Intelligence (CSCSI'96)*, 172–186.

F. Lackinger and W. Nejdl (1991). Integrating Model-Based Monitoring and Diagnosis of Complex Dynamic Systems. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, 1123–1128.

H. Levesque, R. Reiter, Y. Lespérance, F. Lin and R. Scherl (1997). GOLOG: A Logic Programming Language for Dynamic Domains. In *Journal of Logic Programming, Special Issue on Actions* **31**(1–3):59–83.

F. Lin and R. Reiter (1994). State Constraints Revisited. *Journal of Logic and Computation* **4**(5):655–678.

F. Lin and R. Reiter (1995). How to Progress a Database II: The STRIPS Connection. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, 2001–2007.

J. Lloyd (1987). *Foundations of Logic Programming* (2nd ed.). Springer Verlag.

S. McIlraith (1994). Further Contributions to Characterizing Diagnosis. *Annals of Mathematics and Artificial Intelligence* **11**(1–4):137–167.

S. McIlraith (1994a). Towards a Theory of Diagnosis, Testing and Repair. In *Proceedings of The Fifth International Workshop on Principles of Diagnosis*, 47–53.

S. McIlraith (1997). Towards a Formal Account of Diagnostic Problem Solving. PhD thesis, Department of Computer Science, University of Toronto, Toronto, Ontario, Canada.

S. McIlraith (1997a). Representing Actions and State Constraints in Model-Based Diagnosis. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, 43–49.

D. Poole (1988). Representing Knowledge for Logic-Based Diagnosis. In *Proceedings of the Fifth Generation Computer System Conference (FGCS-88)*, 1282–1290.

R. Reiter (1987). A Theory of Diagnosis from First Principles. *Artificial Intelligence* **32**:57–95.

R. Reiter (1991). The Frame Problem in the Situation Calculus: A Simple Solution (sometimes) and a Completeness Result for Goal Regression. In *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of J. McCarthy*, 359–380. San Diego, CA: Academic Press.

R. Reiter (1992). The Projection Problem in the Situation Calculus: a Soundness and Completeness Result, with an Application to Database Updates. In *Proceedings First International Conference on AI Planning Systems*, 198–203.

R. Reiter (1996). Natural Actions, Concurrency and Continuous Time in the Situation Calculus. In L. Aiello, J. Doyle and S. Shapiro (ed.) *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR'96)*, 2–13. Cambridge, Massachusetts: Morgan Kaufmann.

R. Reiter (1998). *KNOWLEDGE IN ACTION: Logical Foundations for Describing and Implementing Dynamical Systems.* In preparation.

M. Shanahan (1993). Explanation in the Situation Calculus. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)*, 160–165.

M. Shanahan (1997). Event Calculus Planning Revisited. In *Proceedings 1997 European Conference on Planning (ECP 97), Springer-Verlag Lecture Notes in Artificial Intelligence (no. 1348)*, 390—402.

M. Thielscher (1997). A Theory of Dynamic Diagnosis. In *Linköping Electronic Articles in Computer and Information Science* **2**(11). Research article received for discussion October, 1997. Revised March, 1998. http://www.ep.liu.se/ea/cis/1997/011/.

R. Waldinger (1977). Achieving Several Goals Simultaneously In E. Elcock and D. Michie (ed.) *Machine Intelligence 8*,94–136. Edinburgh, Scotland: Ellis Horwood.

Nonmonotonic Reasoning

Comparing Consequence Relations

Peter A. Flach
Dept. of Computer Science
University of Bristol
Bristol BS8 1UB, UK
Peter.Flach@cs.bris.ac.uk
http://www.cs.bris.ac.uk/~flach/

Abstract

The technical problem addressed in this paper is, given two rule systems for consequence relations **X** and **Y**, how to construct **Y**-approximations of a given **X**-relation. While an upper **Y**-approximation can be easily constructed if all **Y**-rules are Horn, the construction of lower **Y**-approximations is less straightforward. We address the problem by defining the notion of *co-closure under co-Horn rules*, that can be used to remedy violation of certain rules by *removing* arguments. In particular, we show how the co-closure under Monotonicity can be used to construct the monotonic *restriction* of a preferential relation. Unlike the more usual closure under the rules of **M**, this co-closure operator supports the intuition that preferential reasoning is more liberal than monotonic reasoning. The approach is embedded in a general framework for comparing rule systems for consequence relations. A salient feature of this framework is that it is also possible to compare rule systems that are not related by metalevel entailment.

1. INTRODUCTION

1.1 MOTIVATION AND SCOPE

Nonmonotonic reasoning is the process of 'tentatively inferring from given information rather more than is deductively implied' (Makinson, 1994). Nonmonotonic reasoning can thus be said to be more *liberal* than monotonic reasoning. Correspondingly, the set of arguments accepted by a nonmonotonic reasoning agent (also called a *consequence relation*, and defined as a subset of $L \times L$, where L is the language) can be divided into a deductive or monotonic part and a nonmonotonic part. Let us call the function which maps an arbitrary consequence relation to its monotonic core the *monotonic restriction*.

Although the notion of monotonic core has been considered before (Stachniak, 1993), it does not seem to occupy a central place in the theory of nonmonotonic consequence relations, and operators to construct the monotonic core of a given relation have not been defined before.[1] Kraus *et al.* (1990) define a monotonic closure operator, which however maps a consequence relation to a monotonic *superset* (and may therefore be called the *monotonic extension*). The operator seems to be inspired by the Horn form of the rules they consider. However, as we show in this paper even with Horn rules it is possible to apply them in the reverse direction to remove arguments from the consequence relation.

Another aspect we clarify in this paper is the role of metalevel entailment between rule systems. For instance, we have that all the rules of **P** are rules of **M**, hence all monotonic consequence relations are preferential. In our view this is a special case of a more general phenomenon, namely that **P**-semantics encodes more information than **M**-semantics, because it has to distinguish more consequence relations. However, the presence of metalevel entailment does not, by itself, indicate whether this extra information is used to establish a more liberal or rather a less liberal form of reasoning.

Moreover, metalevel entailment is not even a necessary condition for one rule system to be more liberal than another. This will be demonstrated by defining a variant of **P** that is incomparable to it wrt. metalevel entailment (each system includes a rule that is not a rule of the other system), yet clearly and unambiguously axiomatises a less liberal form of reasoning than **P**. In fact, failure to relate these rule systems by an existing comparison criterion was the original motivation for this paper.

1.2 AN EXAMPLE

Consider two reasoning agents NM and CNM, which differ only in the way they handle contradictory information: while NM infers everything from contradictory premisses, CNM refuses to draw any conclusions from them. For all other premisses they agree on the consequences. It follows that the set of CNM-

[1] I.e. operators that work directly on the consequence relation (rather than on its semantic characterisation).

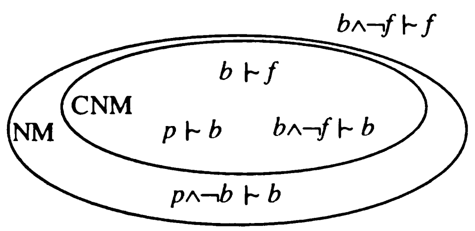


Figure 1. NM is a more liberal reasoner than CNM.

arguments is a subset of the set of NM-arguments (Figure 1). For instance, both NM and CNM infer b from p, but while NM infers anything (including b) from $p\wedge\neg b$, CNM considers those premisses to have no consequences.

Notice that NM and CNM can predict each other's behaviour and hence, in a sense, employ the same information in their reasoning. Specifically, CNM can reconstruct X's behaviour by the rule 'if I don't infer anything from given premisses, NM will infer everything from them; if on the other hand I do infer some consequences, NM will infer exactly the same'. In other words, CNM drops *conclusions* without dropping *information*.

As indicated in Figure 1 NM does not conclude f from $b\wedge\neg f$, i.e. NM considers $b\wedge\neg f$ to be contradictory. Since NM does conclude f from b it follows that NM is a nonmonotonic reasoner. Now consider two other reasoning agents M1 and M2, neither of which accepts an inference from α to β without treating $\alpha\wedge\neg\beta$ as contradictory premisses (from which they, like NM, infer everything). This means that neither M1 nor M2 can reason exactly like NM: if they want to keep the inference from b to f they should, unlike NM, consider $b\wedge\neg f$ contradictory, while if they follow NM in not considering $b\wedge\neg f$ contradictory they should drop the inference from b to f. As it turns out, M1 takes the first option and hence infers everything from $b\wedge\neg f$, while M2 drops the inference from b to f (Figure 2).

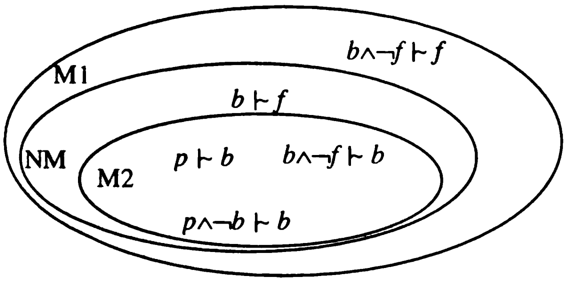


Figure 2. Upper and lower approximations of NM.

Clearly, M1 is strictly more liberal than NM and M2 is strictly less liberal than NM. Furthermore, NM is perfectly able to predict the behaviour of both M1 and M2, but neither M1 nor M2 can exactly reproduce NM's behaviour. M1 cannot predict NM, because NM treats the arguments 'from b infer f' and 'from p infer b' differently, while M1 treats them in the same way. Although M2 and NM agree on what they consider contradictory, M2 does not know for what non-contradictory $\alpha\wedge\neg\beta$ NM accepts the argument 'from α infer β'. In both cases, information has been dropped that cannot be reconstructed. Notice that — compared to NM — M1 drops information to infer *more* conclusions, while M2 drops information to infer *less* conclusions. Also, note that M1 and M2 cannot reproduce each other's behaviour.

In these examples NM embodies the prototypical nonmonotonic reasoner, who is willing to infer f from b by default, at the same time accepting $b\wedge\neg f$ as an exceptional but not contradictory circumstance (the reader may want to read 'it is a bird' for b, 'it flies' for f, and 'it is a penguin' for p — note that the inference from p to b is treated as a deductive inference by all reasoners). In contrast, M1 and M2 are classical monotonic reasoners, who are unable to deal with such default inferences: they either accept the exception $b\wedge\neg f$ as being non-contradictory and drop the default inference (M2), or else reconstruct the default inference as a deductive inference, turning the exception into a contradiction (M1).

It is easy enough to define a closure operator constructing M1 from NM. In this paper we define a co-closure operator constructing M2 from NM. As M2 represents the monotonic core of NM, this operator stays close to the intuition that NM 'jumps to conclusions'. We will also explain why CNM may be considered a more conservative form of reasoning than NM, even though there is no closure operator to map NM to CNM or *vice versa*.

1.3 APPROACH

In this paper we will address the issues mentioned above by introducing a concept of *reduction* that is similar to its counterpart in computational complexity theory. If **X** and **Y** are rule systems, we define a reduction of **X** to **Y** as a function f mapping consequence relations to consequence relations, such that x satisfies the rules of **X** iff $f(x)$ satisfies the rules of **Y**. A reduction establishes a correspondence between **X**-reasoners and **Y**-reasoners, such that any **X**-reasoner can predict the behaviour of the corresponding **Y**-reasoner. This correspondence then establishes a relation between **X** and **Y**; for instance, it may map any **X**-relation to a **Y**-relation that is a subset or superset. It can also be used to investigate the relation between rule systems that are incomparable by metalevel entailment.

The rest of the paper is organised as follows. The formal preliminaries are given in Section 2. Section 3 introduces reductions, and the derived notions of extension and restriction, and applies these to various rule systems. In Section 4 we discuss the main implications of this work.

2. PRELIMINARIES

The formal background of this paper is rooted in the work on abstract consequence relations that are axiomatised by metalevel rules (Gabbay, 1985; Makinson, 1989; Kraus *et al.*, 1990). Kraus, Lehmann and Magidor have characterised several sets of such metalevel rules in their seminal paper (Kraus *et al.*, 1990), the most important of which are **M** for monotonic or deductive reasoning, **P** for preferential reasoning, and **C** for cumulative reasoning. These rule systems are related by metalevel entailment: an axiomatisation of **P** is obtained by adding the rule of Or to **C**, and therefore all rules of **C** are entailed by **P** (see Definition 1 below). Similarly, **M** is axiomatised by **P** augmented with the rule of Monotonicity.

2.1 THE METALANGUAGE

In this section we define the metalanguage used for formulating rule systems. We mostly follow the terminology and notation of (Kraus *et al.*, 1990); readers familiar with that paper may want to skip this section.

Throughout the paper L is a propositional language over a countable set of proposition symbols, closed under the usual logical connectives. We are furthermore given a set of propositional models U, and a satisfaction relation $\models\ \subseteq U\times L$ that is well-behaved with respect to the logical connectives and compact. As usual, we write $\models\alpha$ for $\forall m\in U$: $m\models\alpha$, for arbitrary $\alpha\in L$. Note that U may be a proper subset of the set of all truth-assignments to proposition symbols in L, which would reflect prior knowledge or *background knowledge* of the reasoning agent. Equivalently, we may think of U as the set of models of an implicit background theory T, and let $\models\alpha$ stand for 'α is a logical consequence of T'.

2.1.1 Syntax

The metalanguage for reasoning about consequence relations is a restricted predicate language built up from a unary metapredicate $\models$ in prefix notation (standing for validity with respect to U in L) and a binary metapredicate $\vdash$ in infix notation (standing for an unspecified relation of consequence). In referring to object-level formulae from L we employ a countable set of metavariables α, β, γ, δ, ..., and the logical connectives from L act as function symbols on the metalevel. Metalevel literals are atomic formulae or their negation; instead of $\neg(\models\alpha)$ we write $\not\models\alpha$, and instead of $\neg(\alpha\vdash\beta)$ we write $\alpha\nvdash\beta$. Formulae of the metalanguage, often referred to as *rules* or *properties*, are of the form $P_1,\ldots,P_n\,/\,Q$ for $n\geq0$ (usually written in an expanded Gentzen-style notation), where $P_1,\ldots,P_n$ and Q are literals. Intuitively, such a rule should be interpreted as an implication with *antecedent* $P_1,\ldots,P_n$ (interpreted conjunctively) and *consequent* Q, in which all variables are implicitly universally quantified. A *rule system* is a set of such metalevel rules, denoted by abbreviations in boldface capitals.

2.1.2 Semantics

Consequence relations provide the semantics for this metalanguage, by fixing the meaning of the metapredicate $\vdash$. Formally, a consequence relation is a subset of $L\times L$. They will be used to model part or all of the reasoning behaviour of a particular reasoning agent, by listing a number of *arguments* (pairs of *premiss* and *conclusion*) the agent is prepared to accept. A consequence relation satisfies a rule whenever it satisfies all instances of the rule, and violates it otherwise, where an instance of a rule is obtained by replacing the variables of the rule with formulae from L. A consequence relation satisfies an instance of a rule if, whenever it satisfies the ground literals in the antecedent of the rule, it also satisfies the consequent. A consequence relation satisfies a negated ground literal if it does not satisfy the unnegated ground literal. Finally:

- a ground literal $\models\alpha$ is satisfied whenever the propositional formula from L denoted by α is true in every model in U;
- a ground literal $\alpha\vdash\beta$ is satisfied whenever the pair of propositional formulae from L denoted by α and β is an element of the consequence relation.

It is customary to ignore the distinction between the metalanguage and its semantics by referring to a particular consequence relation as $\vdash$ and writing $p\vdash q$ instead of $\langle p,q\rangle\in\vdash$. If **X** is a rule system, a consequence relation satisfying the rules of **X** is called an **X**-relation. Rule system **X** *entails* rule system **Y** if every **X**-relation is a **Y**-relation.

2.2 RULE SYSTEMS

In this section we introduce the rule systems considered in this paper.

2.2.1 The systems C, P and M

Among the rule systems studied by Kraus *et al.* (1990) are the following.

DEFINITION 1 (Rule systems **C**, **P**, and **M**). The rule system **C** (for cumulative reasoning) consists of the following rules:

Reflexivity: $$\alpha\vdash\alpha$$

Left Logical Equivalence: $$\frac{\models\alpha\leftrightarrow\beta\ ,\ \alpha\vdash\gamma}{\beta\vdash\gamma}$$

Right Weakening: $$\frac{\models\alpha\rightarrow\beta\ ,\ \gamma\vdash\alpha}{\gamma\vdash\beta}$$

Cut: $$\frac{\alpha\vdash\beta\ ,\ \alpha\wedge\beta\vdash\gamma}{\alpha\vdash\gamma}$$

Cautious Monotonicity: $\dfrac{\alpha \mid\!\sim \beta \;,\; \alpha \mid\!\sim \gamma}{\alpha\wedge\beta \mid\!\sim \gamma}$

The rule system **P** (for preferential reasoning) consists of the rules of **C** plus the following rule:

Or: $\dfrac{\alpha \mid\!\sim \gamma \;,\; \beta \mid\!\sim \gamma}{\alpha\vee\beta \mid\!\sim \gamma}$

The rule system **M** (for monotonic reasoning) consists of the rules of **P** plus the following rule:

Monotonicity: $\dfrac{\models\alpha\rightarrow\beta \;,\; \beta \mid\!\sim \gamma}{\alpha \mid\!\sim \gamma}$

The axiomatisations of **C**, **P** and **M** have been chosen such that they can be obtained from one another by adding or deleting rules. Consequently, **M** entails **P** and **P** entails **C**. Note that Cautious Monotonicity and Left Logical Equivalence are redundant in **M**, since they are implied by Monotonicity.

The main result of (Kraus *et al.*, 1990) is a characterisation of these rule systems in terms of the following semantics (with slight changes of terminology):

DEFINITION 2 (Cumulative, preferential and monotonic structures). A *cumulative* structure is a triple $W = \langle S,l,<\rangle$, where S is a set of *states*, l: $S\rightarrow 2^U$ is a function that labels every state with a nonempty set of models, and $<$ is a binary relation[2] on S. A state $s\in S$ satisfies a formula $\alpha\in L$ iff for every model $m\in l(s)$, $m \models \alpha$; the set of states satisfying α is denoted by $[\alpha]$. The consequence relation defined by W is denoted by $\mid\!\sim_W$ and is defined by: $\alpha \mid\!\sim_W \beta$ iff every state minimal (wrt. $<$) in $[\alpha]$ satisfies β.
A *preferential* structure is a cumulative structure $\langle S,l,<\rangle$ where every label $l(s)$ is a singleton, and $<$ is a strict partial order (i.e., $<$ is irreflexive and transitive).
A *monotonic* structure is a preferential structure $\langle S,l,\varnothing\rangle$, i.e. the preference relation is empty.

The intermediate level of states allows the same model to appear at several points in the ordering.

2.2.2 The system CP

In order to capture the behaviour of the reasoning agent CNM from the introduction of this paper, who refuses to draw any conclusion from contradictory premisses, we introduce the following rule system.

[2]$<$ is not necessarily a partial order, but it should satisfy a certain 'smoothness condition', which is for instance satisfied if $<$ does not have infinite descending chains.

DEFINITION 3 (Consistent preferential reasoning). The rule system **CP** consists of the rules of **P** with the exception of Reflexivity, and additionally the following two rules:

Consistent Reflexivity: $\dfrac{\alpha \mid\!\sim \alpha \;,\; \alpha \not\mid\!\sim \neg\beta}{\beta \mid\!\sim \beta}$

Consistency: $\dfrac{\models\alpha\rightarrow\neg\beta}{\alpha \not\mid\!\sim \beta}$

A *consistent preferential* structure is a preferential structure $W = \langle S,l,<\rangle$. The consequence relation defined by W is denoted by $\mid\!\sim_W$ and is defined by: $\alpha \mid\!\sim_W \beta$ iff (*i*) $[\alpha]=\varnothing$, and (*ii*) every state minimal in $[\alpha]$ satisfies β.

Similar forms of reasoning have been considered in the literature before (e.g. Benferhat *et al.*, 1992). The system **CP** is included here mainly for the sake of argument; however, we will briefly pause to comment on one of its possible applications.

Consistent preferential reasoning was studied in (Flach, 1995) as a model for a certain kind of induction called *confirmatory induction*, which is a form of closed-world reasoning based on the assumption 'objects that I haven't seen behave like objects I have seen'. In this form of reasoning $\alpha \mid\!\sim \beta$ is interpreted as 'observations α *confirm* inductive hypothesis β', and the rule of Consistency means that contradictory observations do not confirm any hypotheses. Apart from being intuitively justifiable, this enables the unification of confirmatory induction with *explanatory induction*, where a hypothesis is required to entail the observations unless they are contradictory.

Clearly, in the presence of Consistency, various properties such as Supraclassicality (from $\models\alpha\rightarrow\beta$ derive $\alpha \mid\!\sim \beta$) are too strong; this is remedied by replacing Reflexivity with the weaker rule Consistent Reflexivity. Consequently, **P** and **CP** do not entail each other. Notice that consistent preferential structures consist of the same information as preferential structures, but this information is used in a different way by the addition of condition (*i*). For a proof of the completeness of **CP** with respect to consistent preferential structures see (Flach, 1995; 1996).

2.3 CLOSURES AND CO-CLOSURES

We introduce some new terminology, drawing upon an analogy with logic programming. This analogy is revealed by viewing the formulae from the object language L as ground terms in a Herbrand universe. Consequence relations then correspond to Herbrand interpretations (restricted to the metapredicate $\mid\!\sim$) of the metalanguage, whose rules can be easily transformed to clausal notation.

2.3.1 Closure under Horn rules

DEFINITION 4 (Definite rules, indefinite rules, and denials). A rule $P_1,\ldots,P_n \,/\, Q$ is called

1. *definite* if all of $P_1,\ldots,P_n$ and Q are positive literals;
2. *indefinite* if at least one of $P_1,\ldots,P_n$ is a negative literal and Q is a positive literal;
3. a *denial* if all of $P_1,\ldots,P_n$ are positive literals and Q is a negative literal.[3]

This exhausts all the possibilities: the case that at least one of $P_1,\ldots,P_n$ is a negative literal and Q is a negative literal can be rewritten to case 1 or case 2.

EXAMPLE 1. All of the above rules are definite, except the added **CP**-rules: Consistent Reflexivity is an indefinite rule, and Consistency is a denial.

As is well-known, with a set of definite rules **D** one can associate an immediate consequence operator, which maps a set of arguments A to its immediate consequences under **D**, as follows (ground$_L$(**D**) stands for the set of ground instances of rules in **D** over the Herbrand universe L):

$$T_{\mathbf{D}}(A) = \{Q \mid P_1,\ldots,P_n \,/\, Q \text{ is a definite rule in } \mathrm{ground}_L(\mathbf{D}) \text{ and } P_1,\ldots,P_n \text{ is satisfied by } A\}$$

We will make use of the following proposition, well-known from logic programming theory.

PROPOSITION 1 (Horn closure). *Let **D** be a set of definite rules. The intersection of any set of **D**-relations is also a **D**-relation. The smallest **D**-relation containing a given set of arguments is unique and equal to the intersection of all **D**-relations containing the given arguments, and also to the least fixpoint of the immediate consequence operator $T_{\mathbf{D}}$, starting from the given arguments.* ■

The latter construction is called the ***D**-closure* of the original set of arguments. As a denial does not produce positive consequences, Proposition 1 also holds for sets of definite rules and denials, jointly called *Horn rules*.

Although they don't use the above terminology, Kraus *et al.* define, for each rule system they consider, a corresponding closure operator. Thus, these closure operators use the metalevel rules (which are all Horn) to derive further arguments. For instance, the **M**-closure operator will turn a preferential consequence relation into a monotonic superset. Intuitively, the **M**-closure arises from the assumption that the default rules employed by the preferential reasoner are actually without exceptions.

[3]For determining whether a rule is definite, indefinite or a denial, literals with the 'built-in' predicates $\models$ and $\not\models$ can be ignored.

EXAMPLE 2. Consider Figure 2. NM violates Monotonicity because $b \mathrel{|\!\sim} f$ while $b \wedge \neg f \mathrel{|\!\not\sim} f$. The **M**-closure operator will add $b \wedge \neg f \mathrel{|\!\sim} f$ by virtue of Monotonicity. Furthermore, assuming that NM is a **P**-reasoner we already have $b \wedge \neg f \mathrel{|\!\sim} \neg f$ by Reflexivity and Right Weakening. In a next iteration the **M**-closure operator will therefore add $b \wedge \neg f \mathrel{|\!\sim} f \wedge \neg f$ by virtue of Right And (a derived rule of **M**). Finally, we obtain $b \wedge \neg f \mathrel{|\!\sim} \delta$ for all $\delta \in L$ because of Right Weakening, i.e. $b \wedge \neg f$ is contradictory. Notice that in general it is insufficient to close off under Monotonicity only (see Example 5 for a counter-example).

2.3.2 Co-closure under co-Horn rules

A less common but in the context of this paper very useful dual of the above is obtained if we consider complements of consequence relations, and view $\alpha \mathrel{|\!\not\sim} \beta$ as a 'co-positive' literal and $\alpha \mathrel{|\!\sim} \beta$ as a 'co-negative' literal.

DEFINITION 5 (Co-definite rules, co-indefinite rules, co-denials, and co-Horn rules). A rule $P_1,\ldots,P_n \,/\, Q$ is called

1. *co-definite* if exactly one of $P_1,\ldots,P_n$ is a positive literal, and Q is a positive literal;
2. *co-indefinite* if at least two of $P_1,\ldots,P_n$ are positive literals and Q is a positive literal;
3. a *co-denial* if all of $P_1,\ldots,P_n$ are negative literals and Q is a positive literal;
4. *co-Horn* if it is either co-definite or a co-denial.

EXAMPLE 3. Left Logical Equivalence, Right Weakening, Monotonicity, Consistent Reflexivity and Consistency are co-definite; Cut, Cautious Monotonicity and Or are co-indefinite; and Reflexivity is a co-denial.

We can thus define an immediate co-consequence operator given a set of co-definite rules **CD**, which operates on a set of arguments A and computes the set of immediate co-consequences of A (arguments to be removed from A) under **CD**.

$$CT_{\mathbf{CD}}(A) = \{P \mid \neg P_1,\ldots,\neg P_n, P \,/\, Q \text{ is a co-definite rule in } \mathrm{ground}_L(\mathbf{CD}) \text{ and } \neg P_1,\ldots,\neg P_n \text{ and } \neg Q \text{ are satisfied by } A\}$$

The following proposition is the dual of Proposition 1:

PROPOSITION 2 (Co-Horn co-closure). *Let **CD** be a set of co-definite rules. The largest **CD**-relation contained in a given set of arguments is* *unique and equal to the union of all CD-relations contained in those arguments, and also to the complement of the least fixpoint of the immediate co-consequence operator CT_{CD}, starting from the complement of the given arguments.* ■

The latter construction is called the *CD-co-closure* of the original set of arguments. It is the main technical tool for obtaining the results in the next section.

EXAMPLE 4. Consider again Figure 2. The co-closure of NM under Monotonicity will remove $b \vdash f$, since $b \wedge \neg f \nvdash f$ is satisfied by NM.

3. COMPARING RULE SYSTEMS

We now come to the main part of the paper. Section 3.1 defines reductions between rule systems, and the conditions under which these may establish extensions or restrictions. The relations between **P** and **M** and between **P** and **CP** are studied in Sections 3.2 and 3.3.

3.1 REDUCTIONS, EXTENSIONS AND RESTRICTIONS

We want to characterise the difference in information encoded in rule systems **X** and **Y**, or equivalently in the semantics characterising them. Generally speaking, a semantics for **X** has two purposes:

1. to distinguish between different **X**-relations, and
2. to distinguish between **X**-relations and non-**X**-relations, i.e. to answer the decision problem 'is x an **X**-relation?'

The idea of a reduction is to find a rule system **Y** and a mapping f such that this latter decision problem is equivalent to the decision problem 'is $f(x)$ a **Y**-relation?' (Figure 3). We may lose the distinction between some of the **X**-relations in the process, in which case the **X**-semantics encodes more information than the **Y**-semantics. Assuming that all **Y**-relations are images under f (otherwise the **Y**-semantics would need additional information to characterise those **Y**-relations not in the co-domain of f), we may say that f encodes the difference in information encoded in **X** and **Y**.

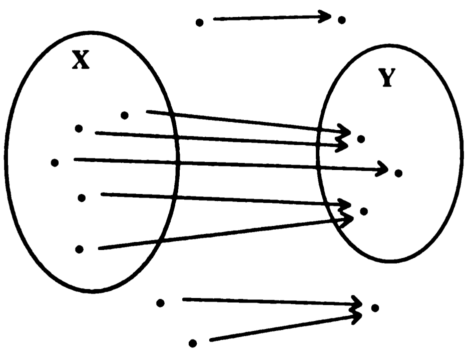


Figure 3. A reduction of **X** to **Y**.

DEFINITION 6 (Reduction). Given two rule systems **X** and **Y**, a *reduction* of **X** to **Y** is a function f mapping consequence relations to consequence relations, such that (i) x is an **X**-relation iff $f(x)$ is a **Y**-relation; (ii) every **Y**-relation is the f-image of an **X**-relation. If such a mapping exists we say that **X** *reduces to* **Y**. If in addition **Y** reduces to **X**, we say that **X** and **Y** are *reduction-equivalent*, otherwise **X** *properly* reduces to **Y**.

Notice that the relation 'reduces to' is a pre-order (it is reflexive and transitive).

3.1.1 Reductions between Horn systems

The **M**-closure as defined by Kraus *et al.* is not a reduction of anything else than the empty set of rules to **M**, since it maps *any* consequence relation into a monotonic superset. In general, a reduction of **X** to **Y** must be strong enough to transform **X**-relations into **Y**-relations, but not so strong that it transforms non-**X**-relations into **Y**-relations. Clearly, the **M**-closure is too strong in this sense. There is, however, a way out by taking the *difference* between the **P**-closure and the **M**-closure.

THEOREM 3 (Horn reduction). *Let **X** and **Y** be two rule systems, such that **Y** entails **X**. If every rule of **X** and **Y** is Horn, then **X** reduces to **Y**; if in addition **X** does not entail **Y** the reduction is proper.*

Proof. If **X** and **Y** are Horn, then the closure of $\vdash$ under **X** and **Y** is well-defined and denoted by $\vdash_X$ and $\vdash_Y$, respectively. Consider the following function:

$$f(\vdash) = \vdash_Y - (\vdash_X - \vdash)$$

We will prove that f establishes a reduction of **X** to **Y**. If $\vdash$ is an **X**-relation then $\vdash = \vdash_X$ and thus $f(\vdash) = \vdash_Y$. On the other hand, if $\vdash$ violates a rule of **X** because $\alpha \nvdash \beta$ and $\alpha \vdash_X \beta$, then the inference is removed from $\vdash_Y$ and thus $f(\vdash)$ violates the same rule of **X**. Clearly f maps onto the whole of **Y** because **Y**-relations are mapped onto itself.

If **X** does not entail **Y** there are more **X**-relations than **Y**-relations, hence there is no reduction of **Y** to **X**. ■

Notice that both **X** and **Y** are required to be Horn — we cannot use the construction in the proof of Theorem 3 to reduce a non-Horn system to a Horn system.

3.1.2 Extensions and restrictions

Once we have established a reduction of **X** to **Y**, we may want to investigate the relation between **X**-relations and the **Y**-relations they are mapped to.

DEFINITION 7 (Extension and restriction). Given a reduction of **X** to **Y**, its restriction to the set of **X**-relations is called a *semi-reduction* of **X** to **Y**. A semi-reduction is an *extension* (*restriction*) if it maps every consequence relation to a superset (subset); we say that **X** *extends to* (*restricts to*) **Y**. The extension (restriction) is *proper* if in addition **X** properly reduces to **Y**.

The relations 'extends to' and 'restricts to' are partial orders, but **X** may both extend and restrict to **Y** (we will see below that this is the case for **P** and **M**).

$$\begin{tikzcd}
\alpha \vdash \beta \arrow[d, "\text{Monotonicity}"'] \arrow[dr, "\text{Right Weakening}"] & \\
\gamma \vdash \beta \arrow[dr, "\text{Right Weakening}"'] & \alpha \vdash \delta \arrow[d, "\text{Monotonicity}"] \\
& \gamma \vdash \delta
\end{tikzcd}$$

(a) Right Weakening

$$\begin{tikzcd}
\alpha \vdash \delta \arrow[d, "\text{Monotonicity}"'] \arrow[dr, "\text{Or}"] & & \beta \vdash \delta \arrow[dl] \arrow[ddl] \\
\gamma \vdash \delta \arrow[dr] & \alpha \vee \beta \vdash \delta \arrow[d, "\text{Monotonicity}"] & \\
& \gamma \vee \beta \vdash \delta \arrow[u, phantom, "\text{Or}"', shift right=4] &
\end{tikzcd}$$

(b) Or

$$\begin{tikzcd}
\alpha \vdash \beta \arrow[d, "\text{Monotonicity}"'] \arrow[dr, "\text{Cut}"] & & \alpha \wedge \beta \vdash \delta \arrow[dl] \arrow[d, "\text{Monotonicity}"] \\
\gamma \vdash \beta \arrow[dr] & \alpha \vdash \delta \arrow[d, "\text{Monotonicity}"] & \gamma \wedge \beta \vdash \delta \arrow[dl] \\
& \gamma \vdash \delta \arrow[u, phantom, "\text{Cut}"', shift right=4] &
\end{tikzcd}$$

(c) Cut

Figure 4. Confluence of Monotonicity with rules of **P**.

COROLLARY 4. ***C*** *properly extends to* ***P****, and* ***P*** *properly extends to* ***M****.*
Proof. We can use Kraus *et al.*'s **P**-closure as an extension of **C** to **P**, and their **M**-closure as an extension of **P** to **M**. ■

As we have argued before, this closure approach establishes a relation between **P** and **M** which is intuitively unsatisfactory because it deems preferential reasoning more conservative than deductive reasoning. In the next section we define a reduction of **P** to **M** that is intuitively more appealing.

3.2 COMPARING P AND M

It is straightforward to obtain a dual to Theorem 3 which relates co-Horn rule systems by means of their co-closures. Such a result would however have limited practical importance, since none of the rule systems considered in this paper are co-Horn. However, note that Monotonicity is a co-definite rule; we will show that the co-closure under Monotonicity yields a restriction of **P** to **M**, without further help of the rules of **P**.

3.2.1 The restriction of P to M

The following Lemma provides the key insight.

LEMMA 5. *The co-closure under Monotonicity of a* ***P****-relation is an* ***M****-relation.*
Proof. Clearly the co-closure under Monotonicity of any consequence relation is a subset that satisfies Monotonicity. We will show that the removal of arguments will not introduce violations of rules of **P**.
For Reflexivity, $\alpha \vdash \alpha$ would be removed if $\models \gamma \to \alpha$ and $\gamma \nvdash \alpha$ for some γ, but this is impossible if $\vdash$ is preferential (use Reflexivity and Right Weakening).
For Right Weakening, suppose $\alpha \vdash \beta$ and $\models \beta \to \delta$. $\alpha \vdash \delta$ would be removed if $\models \gamma \to \alpha$ and $\gamma \nvdash \delta$ for some γ, but then we would also have $\gamma \nvdash \beta$ by Right Weakening, hence $\alpha \vdash \beta$ would be removed, preventing the violation of Right Weakening.
For Or and Cut an analogous argument holds (see Figure 4).
Finally, Left Logical Equivalence and Cautious Monotonicity are implied by Monotonicity. ■

It should be noted that the dual of Lemma 5 does not hold: if we would close off a preferential relation under Monotonicity only, the resulting relation may violate some rule of **P**. Monotonicity by itself does not fully characterise the difference between a **P**-relation and its **M**-extension.

EXAMPLE 5. Consider the preferential structure with states $s<t<u$ such that s satisfies a, b, ¬c and ¬d, t satisfies a, ¬b, c and ¬d, and u satisfies a, b, c and d. We thus have a $\mid\!\sim$ b and c∧b $\mid\!\sim$ d, but a $\not\mid\!\sim$ d, c $\not\mid\!\sim$ b, and c $\not\mid\!\sim$ d. Suppose now $\models$c→a, then closing off under Monotonicity adds c $\mid\!\sim$ b but not c $\mid\!\sim$ d. The resulting relation violates Cut and is therefore not an **M**-relation.

We will now show that there is a reduction of **P** to **M** of which the co-closure under Monotonicity establishes the semi-reduction.

THEOREM 6. ***P*** *properly restricts to* ***M***.
Proof. Let $\mid\!\sim$ be an arbitrary consequence relation, let $\mid\!\sim_P$ denote its **P**-closure, and let $\mid\!\sim_{M'}$ denote the co-closure under Monotonicity of $\mid\!\sim$. Consider the function g defined as follows:

$$g(\mid\!\sim) = \begin{cases} \mid\!\sim_{M'} \text{ if } \mid\!\sim_P = \mid\!\sim \\ \mid\!\sim \text{ otherwise} \end{cases}$$

We will prove that g establishes a reduction of **P** to **M**. If $\mid\!\sim$ satisfies **P** then $\mid\!\sim = \mid\!\sim_P$ and therefore $g(\mid\!\sim) = \mid\!\sim_{M'}$ which satisfies **M** by Lemma 5. On the other hand, if $\mid\!\sim$ violates a rule of **P** then $\mid\!\sim_P \neq \mid\!\sim$, hence $g(\mid\!\sim) = \mid\!\sim$ violates **M**.
Since **P** does not entail **M** there are more **P**-relations than **M**-relations, hence there is no reduction of **M** to **P**.
Finally, we have that **P**-relations are mapped to subsets, which means that the semi-reduction is a restriction. ■

The reduction g in the proof of Theorem 6 is admittedly not very elegant — note however that $g(\mid\!\sim) = \mid\!\sim_{M'}$ — ($\mid\!\sim_P$ — $\mid\!\sim$) doesn't work because co-closure under Monotonicity may remove violations of **P**. In any case, the importance of this result is that we have obtained an alternative way of relating **P** and **M**, by defining the **M**-restriction of a **P**-relation as its co-closure under Monotonicity.

3.2.2 Semantic characterisations

For completeness we also give semantic characterisations of the above semi-reductions of **P** to **M**. The **M**-extension of a **P**-relation is obtained by throwing out every state which represents an exception to a preferential argument (the preference order becoming obsolete in the process).[4]

[4]Similar results have been obtained by (Stachniak, 1993; Benferhat *et al.*, 1996).

THEOREM 7 (Semantic characterisation of extension of **P** to **M**). *Let $\mid\!\sim$ be a preferential relation characterised by the preferential structure $\langle S,l,<\rangle$, and let $\mid\!\sim'$ be characterised by the monotonic structure $\langle S',l,\varnothing\rangle$ with*

$$S' = S - \{s\in S \mid s \text{ satisfies } \alpha\wedge\neg\beta \text{ for some } \alpha \mid\!\sim \beta\}$$

$\mid\!\sim'$ is the ***M****-extension of $\mid\!\sim$.*
Proof. For every argument $\alpha \mid\!\sim \beta$ we have that every state in S' satisfies $\alpha\to\beta$, hence $\mid\!\sim'$ is a superset of $\mid\!\sim$. Since $\mid\!\sim'$ is monotonic and $\mid\!\sim_M$ is the smallest monotonic superset of $\mid\!\sim$, we have $\mid\!\sim' \supseteq \mid\!\sim_M$; we will prove that $\mid\!\sim' \subseteq \mid\!\sim_M$.
Suppose therefore $\alpha \mid\!\sim' \beta$; we will prove that $\alpha \mid\!\sim_M \beta$. If $\alpha \mid\!\sim \beta$ then clearly $\alpha \mid\!\sim_M \beta$; so suppose $\alpha \not\mid\!\sim \beta$, i.e. there exist states in S satisfying $\alpha\wedge\neg\beta$. Since $\alpha \mid\!\sim' \beta$ all such states have been removed from S — that is, for every state $s\in S$ satisfying $\alpha\wedge\neg\beta$ there are $\delta,\varepsilon\in L$ such that $\delta \mid\!\sim \varepsilon$ and s satisfies $\delta\wedge\neg\varepsilon$. Let Δ denote the conjunction $\bigwedge(\delta\to\varepsilon)$ over these $\delta,\varepsilon\in L$, then we have $\alpha\wedge\neg\beta \mid\!\sim \neg\Delta$, hence by a valid **P**-derivation $\alpha \mid\!\sim \Delta\to\beta$. On the other hand we have $\delta \mid\!\sim \varepsilon$ for each of these $\delta,\varepsilon\in L$, therefore **true** $\mid\!\sim$ $\delta\to\varepsilon$ hence **true** $\mid\!\sim \Delta$. Finally, we have $\alpha \mid\!\sim_M \Delta\to\beta$ and $\alpha \mid\!\sim_M \Delta$, and thus $\alpha \mid\!\sim_M \beta$. ■

The **M**-restriction of a **P**-relation is obtained by ignoring the preference order.

THEOREM 8 (Semantic characterisation of restriction of **P** to **M**). *Let $\mid\!\sim$ be a preferential relation characterised by the preferential structure $\langle S,l,<\rangle$, and let $\mid\!\sim'$ be characterised by the monotonic structure $\langle S,l,\varnothing\rangle$. $\mid\!\sim'$ is the* ***M****-restriction of $\mid\!\sim$.*
Proof. As before we denote the **M**-restriction of $\mid\!\sim$ by $\mid\!\sim_{M'}$. Clearly $\mid\!\sim'$ is a subset of $\mid\!\sim$. Since $\mid\!\sim'$ is monotonic and $\mid\!\sim_{M'}$ is the largest monotonic subset of $\mid\!\sim$, we have $\mid\!\sim' \subseteq \mid\!\sim_{M'}$; we will prove that $\mid\!\sim' \supseteq \mid\!\sim_{M'}$.
Suppose therefore that $\alpha \not\mid\!\sim' \beta$; we will prove $\alpha \not\mid\!\sim_{M'} \beta$. If $\alpha \not\mid\!\sim \beta$ then clearly $\alpha \not\mid\!\sim_{M'} \beta$; so suppose $\alpha \mid\!\sim \beta$. Since $\alpha \not\mid\!\sim' \beta$ there exists a state in S (non-minimal in $[\alpha]$) satisfying $\alpha\wedge\neg\beta$. It follows that $\alpha\wedge\neg\beta \not\mid\!\sim \beta$, hence $\gamma \not\mid\!\sim_{M'} \beta$ for every γ such that $\models\alpha\wedge\neg\beta\to\gamma$ due to the construction of the co-closure under Monotonicity — in particular $\alpha \not\mid\!\sim_{M'} \beta$. ■

As a general conclusion we may say that the relation between **P** and **M** is ambiguous (at least on purely formal grounds), since **P** both extends and restricts to **M**. Our intuition that **P** establishes a logic of 'jumping to conclusions' must therefore be rooted in pragmatics. We will return to the issue in Section 4 below.

3.3 COMPARING P AND CP

The relation between **P** and **CP** is of interest, because neither of these rule systems entails the other. We show that they are still comparable within our framework.

THEOREM 9. ***P** restricts to **CP**, and **CP** extends to **P**.*

Proof. A bijection between the set of **P**-relations and the set of **CP**-relations is established by the fact that their semantic structures take the same form. So let $\langle S,l,<\rangle$ be a (consistent) preferential structure defining the **P**-relation $\vdash$ and the **CP**-relation $\vdash'$. These two consequence relations only differ in arguments with premisses α that are unsatisfiable in S: such premisses are uniquely defined by $\alpha \vdash \neg\alpha$ or alternatively $\alpha \nvdash' \alpha$. We can then define the following functions:

$$h_1(\vdash) = \vdash - \{\langle\alpha,\beta\rangle \mid \alpha,\beta \in L \text{ and } \alpha \vdash \neg\alpha\}$$

$$h_2(\vdash') = \vdash' \cup \{\langle\alpha,\beta\rangle \mid \alpha,\beta \in L \text{ and } \alpha \nvdash' \alpha\}$$

It is easy to show that these functions define reductions from **P** to **CP** and from **CP** to **P**, respectively. The corresponding semi-reductions establish a restriction of **P** to **CP** and and extension of **CP** to **P**, respectively. ■

This result unequivocally establishes **P** as a more liberal form of reasoning than **CP**.

4. DISCUSSION

In this paper we have proposed the notion of reducibility between rule systems in order to characterise their difference. A reduction of **X** to **Y**, if it exists, shows that **X**-semantics has more degrees of freedom than **Y**-semantics. It also constructs a '**Y**-approximation' for a given **X**-relation. We have demonstrated that this notion is more general then metalevel entailment or closure operators by applying it to rule systems that don't entail each other.

In our framework the relation between **M** and **P** is inherently ambiguous: by throwing away exceptions to defaults we construct an **M**-extension of a preferential relation, by throwing away the defaults themselves we construct an **M**-restriction. While the latter reduction is the reason for saying that preferential reasoning jumps to conclusions that are not deductively justified, our framework provides no formal reason for preferring the **M**-restriction over the **M**-extension as the canonical reduction of **P** to **M**. This can of course be seen as a shortcoming of our framework, but it seems to be very hard to explain, in a semantics-independent way, why it is more natural to construct a monotonic structure from a preferential one by throwing away the preference order rather than removing exceptional states.

In the literature the emphasis has been on extensions through closure operators. This suggests a tendency to view metalevel rules as uni-directional inference rules, used to expand a given set of arguments (*cf.* the question 'What does a conditional knowledge base entail?' (Kraus *et al.*, 1990; Lehmann & Magidor, 1992)). However, we have shown that, even if a rule like Monotonicity is a definite rule, it may be sometimes more natural to apply its contrapositive. In other words, such rules are not primarily inference rules, but rather rationality postulates constraining reasoning behaviours. Any consequence relation satisfying a particular rule system is considered rational with respect to the reasoning form axiomatised by that rule system. A logically stronger rule system puts more constraints on possible reasoning behaviours — but this doesn't, by itself, indicate whether the extra rules lead to a more or rather less liberal form of reasoning.

Our perspective may also clarify the situation with respect to the rule of Rational Monotonicity, as studied by Lehmann and Magidor (1992).

Rational Monotonicity: $\dfrac{\alpha \nvdash \neg\beta\ ,\ \alpha \vdash \gamma}{\alpha\wedge\beta \vdash \gamma}$

The system **R** consists of **P** extended with Rational Monotonicity. Since Rational Monotonicity is an indefinite rule, there is no **R**-closure operator (there may be several smallest **R**-relations containing a given consequence relation $\vdash$). From the metalevel viewpoint this is a perfectly natural situation: our metarules are rationality postulates, which may be simply too weak to fully prescribe the behaviour of a reasoning agent. On the other hand, from the connective viewpoint such indefiniteness is clearly unsatisfying, and Lehmann and Magidor go at great lengths to define the notion of rational closure (a preferred superset of $\vdash$ satisfying **R**). However, notice that Rational Monotonicity is a co-definite rule, hence we may investigate its co-closure. Now, if Rational Monotonicity were independent of the rules of **P** in the same way as Monotonicity is independent of the rules of **P** (Lemma 5), it would follow that **P** actually *restricts* to **R**, and we could define rational 'closure' of an arbitrary consequence relation as closure under **P** followed by co-closure under Rational Monotonicity. We leave the investigation of this conjecture as future work.

Acknowledgements

When writing this paper I have profited from discussions with Joe Halpern, Daniel Lehmann, John-Jules Meyer, Mark Ryan, and participants of the *Third Dutch/German Workshop on Nonmonotonic Reasoning Techniques and their Applications*, where an earlier version of this paper was presented. Remarks of anonymous reviewers have also been helpful. Part of this work was supported by Esprit IV Long Term Research Project 20237 ILP2.

References

Salem Benferhat, Didier Dubois and Henri Prade (1992). Representing default rules in possibilistic logic. *Proc. 3d Int. Conf. on Principles of Knowledge Representation and Reasoning*, pp.673–684. Morgan Kaufmann.

Salem Benferhat, Didier Dubois and Henri Prade (1996). Beyond counter-examples to nonmonotonic formalisms: a possibility-theoretic analysis. *Proc. 12th Int. Eur. Conf. on Artificial Intelligence*, pp.652–656. John Wiley.

Peter Flach (1995). Conjectures: an inquiry concerning the logic of induction. PhD thesis, Tilburg University.

Peter Flach (1996). Rationality postulates for induction. *Proc. 6th Int. Conf. on Theoretical Aspects of Rationality and Knowledge*, pp.267–281. Morgan Kaufmann.

Dov Gabbay (1985). Theoretical foundations for non-monotonic reasoning in expert systems. In *Logics and Models of Concurrent Systems*, pp.439–457. Springer.

Sarit Kraus, Daniel Lehmann and Menagem Magidor (1990). Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44:167–207.

Daniel Lehmann and Menagem Magidor (1992). What does a conditional knowledge base entail? *Artificial Intelligence*, 55:1–60.

David Makinson (1989). General theory of cumulative inference. *Proc. 2nd Int. Workshop on Non-Monotonic Reasoning*, pp.1–18. Lecture Notes in Artificial Intelligence 346, Springer.

David Makinson (1994). General patterns in nonmonotonic reasoning. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, Vol.3, pp.35–110. Clarendon Press.

Zbigniew Stachniak (1993). Algebraic semantics for cumulative inference operations. *Proc. 11th Nat. Conf. on Artificial Intelligence AAAI-93*, pp.444–449. MIT Press.

SYSTEM JZ
How to build a canonical ranking model of a default knowledge base

Emil Weydert
Max-Planck-Institute for Computer Science
Im Stadtwald, D-66123 Saarbrücken, Germany
emil@mpi-sb.mpg.de

Abstract

We show how to construct for any consistent weighted default knowledge base a canonical quasi-probabilistic model, or $\kappa\pi$-measure, which may then be exploited for default reasoning. This is achieved through a uniform minimal iterated update process which is inspired from Spohn's revision procedure and normality maximization techniques like system Z. The resulting consequence relation, which is also linked to entropy maximization, satisfies Lehmann's rationality postulates, Halpern and Koller's invariance principles, as well as standard inference desiderata, like inheritance to exceptional subclasses.

1 INTRODUCTION

Conditional inference strategies based on normality maximization have been quite popular in the formalization of default reasoning. Examples are rational closure [LM 92], system Z [PEA 90], specificity minimization [BCDLP 93], and elementary hyperentailment/normality maximization [WEY 93,95], which - modulo cosmetic changes - have turned out to be equivalent for finite propositional default sets. Their basic strengths are the natural and transparent semantic foundations, the acceptable computational properties, the construction of a canonical ranked model, as well as the satisfaction of some standard desiderata, like specificity or defeasible chaining. A serious drawback, however, is their inability to handle implicit independence assumptions, in particular the violation of inheritance to exceptional subclasses.

There are at least two ways to attack this problem. One possibility, reminiscent of more traditional approaches, is to use sophisticated prioritization techniques, like conditional entailment [GP 92]. However, the ad hoc handling of independencies in such purely preference-based approaches leaves us without a proper semantic underpinning, i.e. without even a modest guarantee for correct behaviour. The other possibility is to exploit a well-established, proven methodology, and to take a probabilistic viewpoint. The resulting additional complexity is largely compensated by the better grasp of the dependency issue. Accordingly, in recent years, probabilistic and quasi-probabilistic approaches to default reasoning have become increasingly popular [PEA 89]. Semantic transparency, intuitively fitting results and the opportunity to exploit significant portions of the large and sophisticated tool set of classical probability and decision theory have backed this trend.

The probabilistic route to defeasible inference offers two distinct but related strategies for representing default knowledge. The first one is to refine the classical probabilistic framework and to model defaults through infinitesimal [LM 92, WEY 95] or asymptotically parametrized conditional probability constraints [GMP 90, BGHK 97]. The second one is to adopt, right from the beginning, a coarse, semi-qualitative perspective and to try to express defaults directly by constraints on ranking measures, a kind of order-of-magnitude probability distributions [SPO 88, PEA 90, BCDLP 93, WEY 91, 94]. But of course, this constitutes only the first step. The main task consists in finding appropriate default inference notions exploiting these semantic paradigms. The obvious starting point here are the standard probabilistic reasoning strategies.

A well-known and very powerful probabilistic inference technique is entropy maximization (ME). It amounts to prefer those measures with the highest entropy, i.e. the least information content (w.r.t. the uniform prior distribution). If we want to pick up a single most plausible representative from a closed convex set of probability distributions, there are several results indicating that ME might be considered "optimal" [SJ 80, JAY 83, PV 90], as long as additional information - beyond elementhood in the given set - is not taken into account.

There have been several attempts to exploit entropy maximization for default reasoning, for instance ME for parametrized probability measures [GMP 90], the random worlds account [BGHK 97], and ME for nonstandard distributions, i.e. admitting infinitesimal probabilities [WEY 95]. All these approaches agree on finite propositional default sets, where they usually produce reasonable results. Nevertheless, in the context of default inference, entropy maximization appears to be a rather opaque and cumbersome numerical procedure, difficult to handle in practice, which is computationally and conceptually less appealing than constructive prioritization techniques like system Z, conditional entailmen, and their relatives. Furthermore, the necessity to translate inherently vague statements like defaults into specific hyperprecise - with infinitesimal bounds - conditional probability constraints, and the resulting sensitivity to seemingly irrelevant modifications of the default knowledge base, constitute a serious drawback.

What we are looking for is a sufficiently natural, transparent, and constructive - de jure or de facto - counterpart of ME within the ranking measure framework, which would combine a canonical Z-like construction with an ME-style quasi-probabilistic justification. With system Z* [GMP 90], this has only been achieved for the very restricted class of minimal core sets, i.e. default knowledge bases where the corresponding set of material implications is logically irredundant. We are going to extend these results and present an adventurous - verifying the full set of Lehmann's rationality postulates - but well-motivated and representation independent direct implementation or approximation of the minimal information paradigm within a semi-qualitative, quasi-probabilistic context.

Put in a nutshell, the idea is to adopt a dynamic epistemic perspective. Starting from the observation that ranking measures may be used to represent epistemic valuations, we construct, level by level, through a uniform minimal iterated revision process inspired by Spohn's update procedure for κ-rankings [SPO 88], a canonical ranking measure model of any given finite consistent graded default knowledge base.

We start with the $\kappa\pi$-measure semantics for graded default conditionals and describe a Spohn-type revision strategy for these valuations. Next we introduce our general nonmonotonic inference framework and discuss the exceptional inheritance paradox. It motivates a particular default reasoning philosophy which we call the minimal epistemic construction paradigm. Within this context, we are going to implement our maximal uniformity and minimal update principles to define and justify a new default entailment concept, system JZ. We investigate the inferential features of this approach and illustrate it with several examples. To conclude, we take a look at its relation with ME-based default inference and other competing proposals.

2 RANKING MEASURE MODELS

Ranking measures are coarse-grained probability-like valuations $R : \mathcal{B} \rightarrow \mathcal{V}$ measuring the degree of implausibility (disbelief, surprise) of propositions or world sets. They were introduced in [WEY 91,94] as a natural semantic tool for interpreting default knowledge. According to the corresponding quasi-probabilistic default philosophy, a necessary - although not necessarily sufficient - truth or acceptance condition for default conditionals $\varphi \Rightarrow \psi$ is $R(\neg\psi|\varphi) > 0$, that is $R(\varphi \wedge \psi) < R(\varphi \wedge \neg\psi)$ or $R(\varphi \wedge \neg\psi) = \infty$. In this paper, where we are dealing with finite propositional default knowledge bases, we may restrict ourselves to standard $\kappa\pi$-ranking measures, i.e. whose range consists of the positive reals together with infinity.

Definition 2.1 (Standard $\kappa\pi$-measures)
The map $R : \mathcal{B} \rightarrow \mathcal{R}^+_\infty$ is called a standard $\kappa\pi$-ranking measure on $\mathcal{B}$ iff

- *$\mathcal{B} \subseteq 2^{\mathcal{W}}$ is a boolean set system on $\mathcal{W} \neq \emptyset$,*
- *$\mathcal{R}^+_\infty = ([0,\infty], 0, \infty, +, <)$,*

  with $x \leq \infty + x = x + \infty = \infty$.
- *$R(A \cup B) = Min\{R(A), R(B)\}$,*

  $R(\mathcal{W}) = 0$, $R(\emptyset) = \infty$.
- *If $\forall i \in I$, $R(A_i) = \infty$,*

  then $R(\bigcup\{A_i \mid i \in I\}) = \infty$ (coherence).

The corresponding standard conditional $\kappa\pi$-measure $R(\ |\ ) : \mathcal{B} \times \mathcal{B} \rightarrow \mathcal{R}^+_\infty$ is given by

- *$R(B|A) = R(B \cap A) - R(A)$, for $R(A) \neq \infty$,*

  $R(B|A) = \infty$, for $R(A) = \infty$.

A $\kappa\pi$-measure R is said to be continuous iff

- $R(\bigcup\{A_i \mid i \in I\}) = Inf\{R(A_i) \mid i \in I\}$.

R_0 is the uniform $\kappa\pi$-measure on $\mathcal{B}$, i.e. $R_0(A) = 0$ for all $A \neq \emptyset$.

The $\kappa\pi$-measure concept subsumes the κ- and π-ranking notions advocated in [SPO 88, PEA 90] and [BCDLP 93] for iterated belief revision or default reasoning. κ-measures are just standard $\kappa\pi$-measures taking only integer values and verifying continuity. However, it turns out that some default entailment strategies, in particular system JZ, require rational non-integer values. π-measures or possibility measures are isomorphic to continuous standard $\kappa\pi$-measures as long as we assume probability-like (multiplicative) conditioning and values in the real - not rational - unit interval.

We slightly prefer the $\kappa\pi$-measure framework because it is more general, because our work stands in the tradition of the κ-calculus, and because we want to avoid fuzzy-theoretic connotations and possible confusions with probability values. Another important point is that, for most purposes, $\kappa\pi$-measures may be restricted to rational - although not integer - values. However, because the rational numbers are not closed under $\sqrt{x}$, rational-valued $\kappa\pi$-measures usually do not correspond to rational-valued possibility measures, even if we assume finiteness and probability-like conditioning.

$\kappa\pi$-measures can be interpreted as order of magnitude probabilities in the following sense. Let $P : \mathcal{B} \to [0, 1]$ be any nonstandard probability distribution, i.e. admitting infinitesimal values ε $(0 < \varepsilon < \ldots < 1/3 < 1/2 < 1)$. Then, if P is coherent (as above), we may associate with P a $\kappa\pi$-measure R_P, where $R_P(A)$ is defined to be the standard part - i.e. the closest real number, if it is bounded, and ∞, otherwise - of $\log_\varepsilon(P(A))$.

We are now going to exploit these quasi-probabilistic valuations to interpret propositional default conditionals. Let $\mathcal{L}$ be a language of propositional logic with countably infinitely many propositional variables $P_0, P_1, P_2, \ldots$. To obtain a more fine-grained representation of default knowledge in the context of the $\kappa\pi$-measure framework, for each rational number $a > 0$ or $a = \infty$ (to keep the language countable), we introduce a graded default conditional $\Rightarrow^a$. If $a = 1$, we may drop the index. The default $\varphi \Rightarrow^a \psi$ is roughly intended to mean "*if φ, then we may normally expect ψ with strength a*". $\mathcal{L}(\Rightarrow) = \{\varphi \Rightarrow^a \psi \mid \varphi, \psi \in \mathcal{L}, 0 < a\}$ is the corresponding countable conditional language.

Let $\mathcal{W} = 2^N$ be the set of all $\mathcal{L}$-interpretations (worlds), $\models \subseteq \mathcal{W} \times \mathcal{L}$ be the classical satisfaction relation with $w \models P_i$ iff $w(i) = 1$, and $\vdash$ be the corresponding entailment relation over $\mathcal{L}$. For each $\Sigma \subseteq \mathcal{L}$, we set $\mathcal{W}(\Sigma) = \{w \in \mathcal{W} \mid \forall\varphi \in \Sigma\ w \models \varphi\}$. Furthermore, we abbreviate $\mathcal{W}(\{\varphi\})$ by $\mathcal{W}(\varphi)$ and $R(\mathcal{W}(\neg\varphi)|\mathcal{W}(\psi))$ by $R(\neg\varphi|\psi)$. Also let $\mathcal{B} = \mathcal{B}_{\mathcal{L}} = \{\mathcal{W}(\varphi) \mid \varphi \in \mathcal{L}\}$ and $\mathcal{KP}$ be the set of all standard $\kappa\pi$-measures on $\mathcal{B}$.

Definition 2.2 ($\kappa\pi$-semantics)
The satisfaction relation $\models^{\kappa\pi} \subseteq \mathcal{KP} \times \mathcal{L}(\Rightarrow)$ between $\kappa\pi$-measures $R \in \mathcal{KP}$ and sentences $\varphi \Rightarrow^a \psi \in \mathcal{L}(\Rightarrow)$ is defined by

- $R \models^{\kappa\pi} \varphi \Rightarrow^a \psi$ *iff* $R(\neg\varphi|\psi) \geq a$.

For $\Delta \subseteq \mathcal{L}(\Rightarrow)$, we set $\mathcal{KP}(\Delta) = \{R \in \mathcal{KP} \mid R \models^{\kappa\pi} \Delta\}$. Let $\vdash^{\kappa\pi}$ be the monotonic entailment relation over $\mathcal{L}(\Rightarrow)$ induced by $\models^{\kappa\pi}$, i.e. $\Delta \vdash^{\kappa\pi} \delta$ iff $\mathcal{KP}(\Delta) \subseteq \mathcal{KP}(\delta)$.

3 RANKING MEASURE UPDATES

$\kappa\pi$-measures may be seen as epistemic valuations quantifying the degree of disbelief in propositions, or alternatively, the degree of belief in their complements. For instance, $R(\neg A) \geq a$ or $R \models^{\kappa\pi} \mathbf{T} \Rightarrow^a A$ may mean that A is believed to degree a at least. In what follows, we are going to be particularly interested in $\kappa\pi$-measure updates, which are at the core of our default reasoning strategies.

Within the quasi-probabilistic minimal change paradigm, the canonical approach for revising ranking measures with constraints of the form $R(\neg A) \geq a$, i.e. strengthening belief in A to the entrenchment level a at least, is Jeffrey-conditionalization - as far as needed and possible - for $\kappa\pi$-measures [SPO 88,90]. The idea here is to modify R only as much as necessary to satisfy $R(A) \geq a$, which is achieved by uniformly shifting A, and possibly $\neg A$. In fact, this is just what cross entropy minimization, the best-motivated probabilistic update notion [SJ 80], would suggest when translated into the $\kappa\pi$-measure context. For what we have in mind, it is enough to consider consistent revision, i.e. to assume $R(\neg A) = 0$. We simplify the definitions accordingly.

Definition 3.1 (J-conditionalization, as needed)
Let R be a $\kappa\pi$-measure on $\mathcal{B}$, $A \in \mathcal{B}$, $R(\neg A) = 0$ and $a \in [0,\infty]$. Then $R[A \geq a]$ is the unique $\kappa\pi$-measure R^* on $\mathcal{B}$ s.t. $R^*(A) = max\{R(A), a\}$, $R^*(B|\neg A) = R(B|\neg A)$, and $R^*(B|A) = R(B|A)$ if $a < \infty$.

Assuming $R(\neg A) = 0$, we obtain $R[A \geq a]$, the consistent strength-a-revision of R with $\neg A$, by uniformly shifting A upwards - making it less plausible - until $R^*(A)$ reaches $[a,\infty]$. That is, we only change what has to be changed to ensure that $R^*(A) \geq a$. If the constraint is already verified, $R[A \geq a] = R$. For convenience, we also consider a simple instance of L-conditionalization [GOP 96].

Definition 3.2 (L-conditionalization) *$R[A+a]$ is the unique $\kappa\pi$-measure R^* on $\mathcal{B}$ s.t. $R^*(A) = R(A) + a$, $R^*(B|A) = R(B|A)$ if $a < \infty$, and $R^*(B|\neg A) = R(B|\neg A)$.*

The goal of $R[A+a]$ is not to induce a minimal degree of disbelief in A, but to change it by a fixed amount a. It may be interpreted as the aggregation of R with independent evidence of strength a for $\neg A$. L-conditionalization is easier to handle than J-conditionalization because the construction order is irrelevant in consistent contexts. It is easy to see that both notions are closely linked.

Theorem 3.3 (J/L-Equivalence)
Let $\bigcup\{A_i \mid i \leq n\} \neq \mathcal{W}$. Then, for each J-conditionalization-sequence $[A_0 \geq a_0]\ldots[A_n \geq a_n]$, there is a unique sequence of $p_i \in [0, a_i]$ with $R_0[A_0 \geq a_0]\ldots[A_i \geq a_i] = R_0[A_0 + p_0]\ldots[A_i + p_i]$ for each $i \leq n$. Conversely, for every L-conditionalization-sequence $[A_0 + a_0]\ldots[A_n + a_n]$, there is a unique sequence of p_i with $R_0[A_0 + a_0]\ldots[A_i + a_i] = R_0[A_0 \geq p_0]\ldots[A_i \geq p_i]$ for each $i \leq n$.

Given a non-covering set of propositions $S \subseteq \mathcal{B}$, i.e. whose complements may be consistently believed, we are interested in those $\kappa\pi$-measures R on $\mathcal{B}$ which are accessible from the uniform valuation R_0 by (consistent) iterated variable-strength updates $R_0[A_0 + a_0]\ldots[A_n + a_n]$ with $A_i \in S, a_i \geq 0$. This notion of constructibility will be of central importance for our default inference concept.

Definition 3.4 (Constructibility)
Let $\{A_i \mid i \in i\} \subseteq \mathcal{B}$ and $\bigcup\{A_i \mid i \in I\} \neq \mathcal{W}$. A standard $\kappa\pi$-measure R on $\mathcal{B}$ is called constructible over $S = \{A_i \mid i \in I\}$ iff there are a finite subset $\{i_j \mid j \leq n\} \subseteq I$ and $a_j \geq 0$ s.t. $R = R_0[A_{i_0} + a_0]\ldots[A_{i_n} + a_n]$. Let $Constr(S)$ be the set of those $\kappa\pi$-measures which are constructible over S.

4 DEFAULT ENTAILMENT

How may we extract - within the $\kappa\pi$-measure framework - reasonable defeasible conclusions from finite sets of facts $\Sigma \subseteq \mathcal{L}$ and defaults $\Delta \subseteq \mathcal{L}(\Rightarrow)$? For our purposes, the following approach has just the right level of generality.

Definition 4.1 (Preferred model set function)
$\mathcal{I} : 2^{\mathcal{L}(\Rightarrow)} \rightarrow 2^{\mathcal{KP}}$ is called a preferred model set function over $\mathcal{L}(\Rightarrow)$ iff $\mathcal{I}$ associates with every $\Delta \subseteq \mathcal{L}(\Rightarrow)$ a $\kappa\pi$-model set $\mathcal{I}(\Delta)$ s.t.

- *$\mathcal{I}(\Delta) \subseteq \mathcal{KP}(\Delta)$, and*
- *$\mathcal{I}(\Delta) = \emptyset$ implies $\mathcal{KP}(\Delta) = \emptyset$.*

Note that this approach is more general than the abstract framework proposed in [HK 95], because we do not require $\mathcal{I}$ to be globally semantical w.r.t. $\models^{\kappa\pi}$.

Definition 4.2 (Globally semantical)
A preferred model set function $\mathcal{I}$ is called globally semantical w.r.t. $\mathcal{KP}$ iff there is a function $\mathcal{F}$ with $\mathcal{I}(\Delta) = \mathcal{F}(\mathcal{KP}(\Delta))$.

Although this property is certainly desirable, we may want to allow the knowledge base Δ to convey more information than our basic $\kappa\pi$-measure semantics is able to grasp, e.g. implicit independence assumptions encoded in the global syntactic structure of Δ. In fact, as we are going to see, this is even necessary to deal with the so-called exceptional inheritance paradox. But there is another principle, which we do not want to give up.

Definition 4.3 (Locally semantical)
A preferred model set function $\mathcal{I}$ is called locally semantical w.r.t. $\mathcal{KP}$ iff there is a function $\mathcal{F}$ with $\mathcal{I}(\Delta) = \mathcal{F}(\{\mathcal{KP}(\{\varphi \Rightarrow^a \psi\}) \mid \varphi \Rightarrow^a \psi \in \Delta\})$.

That is, only the chosen global set configuration, not the local syntactic representation of the individual default items, should influence the default conclusions.

Every preferred model set function $\mathcal{I}$ defines a default inference relation $\mid\!\sim^{\mathcal{I}} \subseteq 2^{\mathcal{L} \cup \mathcal{L}(\Rightarrow)} \times \mathcal{L}$ for finite knowledge bases $\Sigma \cup \Delta$. By keeping $\Delta \subseteq \mathcal{L}(\Rightarrow)$ fixed, we obtain a corresponding inference relation $\mid\!\sim^{\mathcal{I}}_{\Delta} \subseteq 2^{\mathcal{L}} \times \mathcal{L}$ on $\mathcal{L}$.

Definition 4.4 (Default entailment)
Let $\Sigma \subseteq \mathcal{L}, \Delta \subseteq \mathcal{L}(\Rightarrow)$ be finite, and σ be a conjunction of Σ's elements.

- *$\Sigma \cup \Delta \mid\!\sim^{\mathcal{I}} \psi$ iff $\Sigma \mid\!\sim^{\mathcal{I}}_{\Delta} \psi$ iff $\forall R \in \mathcal{I}(\Delta),\ R(\neg\psi|\sigma) > 0$.*

It is obvious that $\mid\!\sim^{\mathcal{I}}_{\Delta}$ is always a preferential consequence relation in the sense of [KLM 90]. If $\mathcal{I}(\Delta)$ is a singleton set, $\mid\!\sim^{\mathcal{I}}_{\Delta}$ is also rational. In addition, $\mid\!\sim^{\mathcal{I}}$ satisfies the following variant of the defeasible modus ponens principle.

- **DMP** : $\{\varphi\} \cup \{\varphi \Rightarrow^a \psi\} \cup \Delta \mathrel{|\!\sim}^{\mathcal{I}} \psi$.

However, this framework is still too unspecific to enforce desirable inheritance or irrelevance reasoning patterns, like defeasible chaining or inheritance to exceptional subclasses. So, we have to look for suitable specific $\mathcal{I}$. Of particular interest are of course those inference notions which can be characterized in purely $\kappa\pi$-semantical terms, i.e. where $\mathcal{I}$ is globally semantical w.r.t. $\mathcal{KP}$ and $\mathcal{I}(\Delta)$ only depends on $\mathcal{KP}(\Delta)$. Normality maximization [WEY 95], for instance, whose simplest variant picks up the infimum of all $\kappa\pi$-models, is globally semantical.

Definition 4.5 (Normality maximization)
Normality maximization entailment $\mathrel{|\!\sim}^{NM}$ *is defined by the preferred model set function*

$\mathcal{I}_{NM}(\Delta) = \{Inf(\mathcal{KP}(\Delta))\}$.

Unfortunately, these simple-minded semantic-based approaches cannot reasonably deal with some notorious desiderata like the following general version of inheritance to exceptional subclasses.

Inheritance to exceptional subclasses (EI)
For logically independent $\varphi, \psi, \psi' \in \mathcal{L}$,

- $\{\varphi, \neg\psi\} \cup \{\varphi \Rightarrow \psi, \varphi \Rightarrow \psi'\} \mathrel{|\!\sim} \psi'$.

This fundamental pattern doesn't present any problems for most traditional, rule-oriented default formalisms, like Reiter's default logic. However, it turns out to be a slingshot for all purely semantic-based conditional approaches to default reasoning. These are discarded by the following impossibility theorem, which has been presented in a slightly different form in [WEY 95]. It immediately follows from the fact that $\mathcal{KP}(\{\varphi \Rightarrow \psi, \varphi \Rightarrow \psi'\}) = \mathcal{KP}(\{\varphi \Rightarrow \psi, \varphi \Rightarrow (\psi \leftrightarrow \psi')\})$.

Theorem 4.6 (Exceptional inheritance paradox)
Let $\mathcal{I}$ *be a preferred model set function over* $\mathcal{L}(\Rightarrow)$ *which is globally semantical and validates EI. Then, for logically independent* $\varphi, \psi, \psi' \in \mathcal{L}$, *we have*

- $\{\varphi, \neg\psi\} \cup \{\varphi \Rightarrow \psi, \varphi \Rightarrow \psi'\} \mathrel{|\!\sim}^{\mathcal{I}} \mathbf{F}$.

This result trivializes plausible inference from exceptional premises like $\varphi \wedge \neg\psi$. Possible ad hoc solutions, like syntactic restrictions - e.g. stating EI only for φ, ψ, ψ' with disjoint variable sets - or the requirement of explicitly stated independence assumptions [BDP 94] - extending the language accordingly and forcing the user to do most of the work himself - strongly conflict with the original intentions and intuitions of defeasible inference. And of course, we do not only want to realize EI, but also to exploit implicit independence assumptions in general, and in a systematic way. For us, this result clearly shows that constraints on ranking measures cannot fully grasp the intended meaning of defaults, because they prevent an intuitively correct handling of implicit independence assumptions.

5 REVISION ENTAILMENT

Our strategy for extracting additional meaningful information from a default knowledge base Δ is to interpret defaults in a dual way. On one hand, following the static reading, they define the set $\mathcal{KP}(\Delta)$ of admissible epistemic $\kappa\pi$-models. But on the other hand, they also indicate or reflect which epistemic processes, i.e. evidential aggregation or revision steps, may have built up these models. This more dynamic reading of Δ sees defaults not only as ranking measure constraints but also as individual evidence items or evidence declarations guiding iterated update processes.

Let $\Delta = \{\varphi_i \Rightarrow^{a_i} \psi_i \mid i \leq n\} \subseteq \mathcal{L}(\Rightarrow)$ be a finite set of defaults and $\Delta^{\rightarrow} = \{\varphi_i \rightarrow \psi_i \mid i \leq n\}$ be the corresponding set of material implications over $\mathcal{L}$. Then each $\kappa\pi$-model R of Δ represents a belief state which supports every $\varphi_i \rightarrow \psi_i \in \Delta^{\rightarrow}$ with strength a_i at least, i.e. $R(\varphi_i \wedge \neg\psi_i) \geq a_i > 0$. Note that, because $\Delta^{\rightarrow} \vdash \psi$ iff $R(\neg\psi) > 0$, we may interpret $\Delta^{\rightarrow}$ as an explicit propositional belief set associated with the epistemic measure R.

The basic idea behind our epistemic construction philosophy is to assume that the explicit beliefs $\varphi_i \rightarrow \psi_i \in \Delta^{\rightarrow}$ have been added successively with appropriate strengths p_i so as to let the resulting $\kappa\pi$-measure $R = R_0[\varphi_0 \wedge \neg\psi_0 + p_0] \ldots [\varphi_n \wedge \neg\psi_n + p_n]$ eventually validate Δ.

Based on this interpretation, we are now going to present a nonmonotonic entailment strategy exploiting constructibility and allowing inheritance to exceptional subclasses. The idea is to prefer those epistemic $\kappa\pi$-models R of Δ which result from a (minimal) iterated update process starting with the uniform measure R_0 and proceeding by variable strength revision with the explicit implications in $\Delta^{\rightarrow}$, i.e. by shifting their complements. That is, we concentrate on those $\kappa\pi$-models which are constructible over the exceptional proposition $\varphi_i \wedge \neg\psi_i$ of Δ. The epistemic construction paradigm for default reasoning is based on the following minimal requirement for preferred choice functions.

Epistemic Construction principle
A preferred model set function $\mathcal{I}$ for $\kappa\pi$-measures should satisfy $\mathcal{I}(\Delta) \subseteq \mathcal{I}_J(\Delta)$, where $\mathcal{I}_J(\Delta) = \mathcal{KP}(\Delta) \cap$ Constr(S_Δ) and $S_\Delta = \{\mathcal{W}(\varphi \wedge \neg\psi) \mid \varphi \Rightarrow^a \psi \in \Delta\}$.

It follows from the existence of JZ-models for consistent finite Δ, which we are going to construct in the next section, that $\mathcal{KP}(\Delta) \neq \emptyset$ implies $\mathcal{I}_J(\Delta) \neq \emptyset$. Consequently, $\mathcal{I}_J$ is a preferrred model set function and we may use it to define the most unspecific inference concept based on the epistemic construction paradigm. This approach was first discussed, in a slightly different formal framework, in [WEY 96].

Definition 5.1 (J-entailment)
The inference notion $\mid\!\sim^J = \mid\!\sim^{\mathcal{I}_J}$ is called J-entailment.

The main advantages of J-entailment are its simplicity, its robustness, and its validation of the standard inference patterns, among them inheritance to exceptional subclasses. In fact, because $\mathcal{I}_J(\Delta) = \mathcal{I}_J(\{\varphi \Rightarrow \psi, \varphi \Rightarrow \psi'\}) = \{R_0[\varphi \wedge \neg\psi + p][\varphi \wedge \neg\psi' + q] \mid 1 \leq p, q\}$, it follows for $R \in \mathcal{I}_J(\Delta)$ that $R(\neg\psi' | \varphi \wedge \neg\psi) > 0$, i.e. $\{\varphi, \neg\psi\} \cup \{\varphi \Rightarrow \psi, \varphi \Rightarrow \psi'\} \mid\!\sim^J \psi'$.

On the other hand, J-entailment also has some major disadvantages. In fact, with its many degrees of freedom - recall that constructibility is the only preference criterion - and the resulting excessive cautiousness, it may also block some desirable conclusions. The main problem is that this epistemic construction approach doesn't attempt to minimize the update process beyond restricting the propositional building blocks to $S_\Delta = \{\varphi \wedge \neg\psi \mid \varphi \Rightarrow^a \psi \in \Delta\}$. So we are looking for ways to strengthen the epistemic construction principle by a minimal construction requirement.

Consider for instance $\Delta = \{\mathbf{T} \Rightarrow \varphi, \mathbf{T} \Rightarrow \varphi \vee \psi\}$. Here, we have $\mathcal{I}_J(\Delta) = \{R_0[\neg\varphi + p][\neg\varphi \wedge \neg\psi + q] \mid p \geq 1, q \geq 0\}$. A first step towards implementing minimality would be to set $q = 0$, because any admissible shifting of $\neg\varphi$ automatically guarantees the satisfaction of the second default as well. A second step would be to do normality maximization and to choose the minimal p. The "minimally constructible" model therefore appears to be $R_0[\neg\varphi + 1]$, which in our example turns out to be the absolute minimum of $\mathcal{I}_J(\Delta)$.

But note that being minimal in $\mathcal{I}_J(\Delta)$ is neither necessary nor sufficient for what we understand by minimal constructibility. The main idea is rather to shift propositions only insofar as these shiftings turn out to be justified when looking at the resulting full $\kappa\pi$-construction. In particular, the amount of unmotivated, redundant shifting should be minimized. That is, we want to focus on those models which are sufficiently grounded or justified. This is made more precise by the following definition.

Definition 5.2 (Justifiable constructibility)
A $\kappa\pi$-model R of $\Delta = \{\varphi_i \Rightarrow^{a_i} \psi_i \mid i \leq n\}$ is called justifiably constructible over Δ iff $R = R_0[\varphi_i \wedge \neg\psi_i + p_i \mid i \leq n]$ and for each $0 < p_i$, $R(\neg\psi_i|\varphi_i) = a_i$. Let $\mathcal{I}_{JJ}(\Delta)$ be the set of all justifiably constructible models over Δ.

Within a justifiable construction, the proposition $\varphi \wedge \neg\psi$ may only be shifted upwards if some corresponding $\kappa\pi$-measure constraint $R(\neg\psi_i|\varphi_i) \geq a_i$ - with $\varphi_i \wedge \neg\psi_i$ being logically equivalent to $\varphi \wedge \neg\psi$ - is eventually minimally realized, i.e. as an equality constraint. If, on the other hand, its satisfaction is a side-effect of other shifts, $\varphi \wedge \neg\psi$ shouldn't be moved itself. There shouldn't be any unmotivated, redundant shifts. In the example above, $R_0[\neg\varphi + 1]$ is the only justifiably constructible $\kappa\pi$-model. Note however that a default base Δ may well possess a whole family of justifiably constructible models. Because, as before, it can be shown that $\mathcal{KP}(\Delta) \neq \emptyset$ implies $\mathcal{I}_{JJ}(\Delta) \neq \emptyset$, we get another default entailment concept, still quite robust, but considerably stronger than $\mid\!\sim^J$.

Definition 5.3 (JJ-entailment)
The inference notion $\mid\!\sim^{JJ} = \mid\!\sim^{\mathcal{I}_{JJ}}$ is called JJ-entailment.

JJ-entailment is of particular interest because of its close links to robust variants of ME-based default inference. This relationship is discussed in greater detail in a companion paper to this one [WEY 98]. Because $\mathcal{I}_{JJ}$ usually picks up many preferred models, for most Δ, $\mid\!\sim^{JJ}_\Delta$ will not satisfy rational monotony. However, for some purposes, e.g. decision theoretic considerations, a canonical $\kappa\pi$-model might be preferable.

There are two ways to obtain such a model of Δ. The simplest one is to consider $\mathcal{I}^{JJR}(\Delta) = \{\mathrm{Inf}(\mathcal{I}^{JJ}(\Delta))\}$, i.e. $\mid\!\sim^{JJ}$ followed by normality maximization (taking the infimum of all justifiably constructible models). Unfortunately, the resulting model itself may not be constructible over Δ and precious independence information may be lost by taking the infimum. In what follows, we are therefore going to take another road and propose a canonical Z-like shifting construction.

6 JZ-CONSTRUCTION

In this section, we are going to set the foundations for a new entailment concept aimed at combining the minimal epistemic construction paradigm, as implemented by system JJ, with the philosophy of rational closure, as implemented by system Z. Suppose, we have a finite, consistent set of parametrized defaults

- $\Delta = \{\varphi_i \Rightarrow^{a_i} \psi_i \mid i \leq n\} \subseteq \mathcal{L}(\Rightarrow)$.

Each default $\varphi_i \Rightarrow^{a_i} \psi_i$ induces a $\kappa\pi$-measure constraint $R(\varphi_i \wedge \psi_i) + a_i \leq R(\varphi_i \wedge \neg\psi_i)$ and enacts a permission to shift $S_i = \varphi_i \wedge \neg\psi_i$. The idea is to construct, through a minimal and uniform iterated update process, a canonical $\kappa\pi$-model of Δ, which will be called the JZ-model of Δ and be denoted by

- $\mathrm{JZ}[\Delta] = R_0[S_i + p_i \mid i \leq n]$,

where $0 \leq p_i$. An important ingredient of this model construction process is relative normality maximization. Assuming existence, it associates with every $\kappa\pi$-measure R and default set Δ the minimal $\kappa\pi$-model R' of Δ above R. More formally, if $\{R' \mid R \leq R' \in \mathcal{KP}(\Delta)\} \neq \emptyset$ ($R \leq R'$ iff for all A, $R(A) \leq R'(A)$), the relative normality maximization of Δ above R is defined by

- $NM(R, \Delta) = \mathrm{Min}\{R' \in \mathcal{KP} \mid R \leq R' \in \mathcal{KP}(\Delta)\}$.

The general structure of the JZ-construction procedure is as follows. By induction, we are going to construct a sequence $(R^j \mid j \leq s+1)$ of $\kappa\pi$-measures with

- $R^0 = R_0$ and $R^{j+1} = R^j[S_i + p_i \mid i \in I_j]$,

where $I_0 \cup \ldots \cup I_s = I = \{i \mid i \leq n\}$ is a partition of the default indices and $(p_i \mid i \leq n)$ a sequence of weakly positive rational numbers. Let $I_{<j} = \bigcup\{I_p \mid p < j\}$ and $\Delta_j = \{\varphi_i \Rightarrow^{a_i} \psi_i \in \Delta \mid i \notin I_{<j}\}$. Δ_j is the set of those constraints which have not yet been taken into account by R^j. If we understand the JZ-construction as an approximation process integrating the constraints level by level, a desirable requirement for the R^j appears to be

- $R^j_{NM} = NM(R^j, \Delta_j) \models^{\kappa\pi} \Delta$.

Although R^j doesn't have to be a model of $\Delta - \Delta_j$, i.e. the closed, already considered constraints, R^j_{NM}, the smallest model above R^j of Δ_j, i.e. the open, not yet considered constraints, should satisfy Δ. Because $R^{s+1} = NM(R^{s+1}, \emptyset) = NM(R^{s+1}, \Delta_{s+1})$, this ensures that

- $R^{s+1} \models^{\kappa\pi} \Delta$.

On a more general level, the JZ-construction strategy is based on three major guiding principles. Their exact meaning is illustrated by the construction algorithm.

1. **Justifiable constructibility.**
   The $\kappa\pi$-construction should be justifiably constructible, there should be no unmotivated shifts. That is, $\mathrm{JZ}[\Delta](\varphi_i \wedge \psi_i) + a_i = \mathrm{JZ}[\Delta](\varphi_i \wedge \neg\psi_i)$ for $p_i > 0$.

2. **Bottom up constraint satisfaction.**
   The weakest still open constraints $\varphi_i \Rightarrow^{a_i} \psi_i$ should be considered first. That is, we should start by trying to shift those S_i for which the lower bounds $R^j_{NM}(\varphi_i \wedge \psi_i) + a_i$ for $\varphi_i \wedge \neg\psi_i$ are minimal.

3. **Uniformity maximization.**
   Shifting should proceed as uniformly as justifiability permits. That is, for each $j \leq n$, $(p_i \mid i \in I_j)$ should be as close to $\vec{0}$ as possible.

The first requirement means that we are looking for a strengthening of system JJ, which is known to be backed by a robust variant of the maximum entropy paradigm for default reasoning. The second principle helps to guarantee that the process is well-founded and that later shiftings will not invalidate constraints satisfied at earlier stages. The third stipulation implements the classical symmetry, no commitment or equal opportunity conceptions. Taken together, these desiderata will guide, if not determine, our epistemic construction procedure

Now, we are ready to give the details of the inductive JZ-construction process. At stage 0, we start with $R^0 = R_0$, $I_{<0} = \emptyset$, $\Delta_0 = \Delta$ and $R^0_{NM} = NM(R_0, \Delta) \models^{\kappa\pi} \Delta$. At stage $j+1$, by induction, we already have obtained $R^j = R_0[S_i + p_i \mid i \in I_{<j}]$, $I_{<j}$, Δ_j, and we know that $R^j_{NM} = NM(R_j, \Delta_j) \models^{\kappa\pi} \Delta$. Now, there are two tasks to achieve. First, we have to determine I_j. Secondly, we must find the appropriate p_i for $i \in I_j$.

Following the bottom up principle, we are going to put into I_j the indices in $I - I_{<j}$ of those constraints whose realization requires the least update efforts. So, let

- $f_j = \mathrm{Min}\{R^j_{NM}(\varphi_i \wedge \psi_i) + a_i \mid i \notin I_{<j}\}$, and
- $I_j = \{i \notin I_{<j} \mid R^j_{NM}(\varphi_i \wedge \psi_i) + a_i = f_j\}$.

Before we are going to propose specific p_i for $i \in I_j$, we have to see whether, or to what extent, the constraints $R(\varphi_i \wedge \psi_i) + a_i \leq R(\varphi_i \wedge \neg\psi_i)$ for $i \in I_{<j+1}$ are preserved when passing from R^j_{NM} to R^{j+1}_{NM}. Let $S_i = \varphi_i \wedge \neg\psi_i$ and $S'_i = \varphi_i \wedge \psi_i$. Because $R^j_{NM} \leq R^{j+1}_{NM}$, it is enough to verify that $R^j_{NM}(S'_i) = R^{j+1}_{NM}(S'_i)$ for $i \in I_{<j+1}$. By induction and definition, we know

that for $h < j+1$ and $i \in I_h$, $R^h_{NM}(S'_i) = f_h - a_i$. Furthermore, for $h \leq h'$ and $p \notin I_{<h}$, $R^{h'}_{NM}(S_p) \geq R^h_{NM}(S_p) \geq f_h > f_h - a_i$. It also follows from our construction strategy that the $R^{h'}_{NM}$ may only change over $\bigcup\{S_p \mid p \notin I_{<h'}\}$. Consequently, because there is no way to modify the value of S'_i at higher levels h', we get $R^{h'}_{NM}(S'_i) = R^h_{NM}(S'_i) = f_h - a_i$. $R^j_{NM} \models^{\kappa\pi} \Delta - \Delta_{j+1}$, by induction, together with $R^{j+1}_{NM} \models^{\kappa\pi} \Delta_{j+1}$, by definition, therefore ensure that $R^{j+1}_{NM} \models^{\kappa\pi} \Delta$.

What remains to be done is to choose the shifting coefficients p_i for $i \in I_j$ in a way which guarantees justifiable constructibility and maximal uniformity. Here again, we proceed by induction. First, we consider the smallest w s.t. for all $i \in I_j$, $NM(R^j[S_i + w \mid i \in I_j], \Delta_{j+1})(S_i) \geq f_j$. Set $w_0 = w$ and $V_0 = \{i \in I_j \mid NM(R^j[S_i + w_0 \mid i \in I_j], \Delta_{j+1})(S_i) = f_j\}$. Given w_p and V_p, we take the smallest w s.t. $NM(R^j[S_i + w_0 \mid i \in V_0]\ldots[S_i + w_p \mid i \in V_p][S_i + w \mid i \in I_j - (V_0 \cup \ldots \cup V_p)], \Delta_{j+1})(S_i) \geq f_j$ for all $i \in I_j$. Set $w_{p+1} = w$ and $V_{p+1} = \{i \in I_j - (V_0 \cup \ldots \cup V_p) \mid NM(R^j[S_i + w_0 \mid i \in V_0]\ldots[S_i + w_p \mid i \in V_p][S_i + w_{p+1} \mid i \in I_j - (V_0 \cup \ldots \cup V_p)], \Delta_{j+1})(S_i) = f_j\}$. Obviously, the w_i are decreasing. The procedure stops if $I_j - (V_0 \cup \ldots \cup V_p)$ becomes empty or $w_p = 0$. The p_i are now determined as follows.

- If $i \in V_m$, then $p_i = w_m$.
- If $i \in I_j - (V_0 \cup \ldots \cup V_p)$, then $p_i = 0$.

That is, we set $R^{j+1} = R^j[S_i + w_0 \mid i \in V_0]\ldots[S_i + w_p \mid i \in V_p]$. Note that for $i \in V_m$ and $p_i = w_m > 0$, we have $R^{j+1}_{NM}(S_i) = f_j$. On the other hand, if $i \in I_j - (V_0 \cup \ldots \cup V_p)$, we get $p_i = 0$, with $R^{j+1}_{NM}(S_i) \geq f_j$. Consequently, we have justifiable constructibility. By keeping the p_i uniform as far as justifiability permits, we also realize a maximal amount of uniformity. R^{s+1} therefore seems to be an adequate candidate for the role of the single most preferred element of $\mathcal{KP}(\Delta)$.

Definition 6.1 (JZ-model)
$JZ[\Delta] = R^{s+1}$ is called the canonical JZ-model of Δ.

It might be interesting to see whether there is an axiomatic characterization of the JZ-model based on some set of plausibility postulates, analogous to what has been achieved for entropy maximization.

7 SYSTEM JZ

The justifiably constructible and maximally uniform $\kappa\pi$-measures resulting from the JZ-construction process over finite consistent Δ can be used to define a powerful new default inference notion over $\mathcal{L}(\Rightarrow)$.

Definition 7.1 (System JZ)
Let $\mathcal{I}_{JZ}(\Delta) = \{JZ[\Delta]\}$. Then the inference relation $\mid\!\sim^{JZ} = \mid\!\sim^{\mathcal{I}_{JZ}}$ is called JZ-entailment.

It immediately follows from our definitions that each $\mid\!\sim^{JZ}_{\Delta}$ satisfies the full set of Lehmann's rationality postulates.

Theorem 7.2 (Rational inference)
Let $\Delta \subseteq \mathcal{L}(\Rightarrow)$ be finite. Then $\mid\!\sim^{JZ}_{\Delta}$ is a rational consequence relation.

$\mid\!\sim^{JZ}$ does not satisfy classical consistency preservation for $\mid\!\sim^{JZ}_{\Delta}$ on $\mathcal{L}$ (Δ consistent), because each contingent φ verifies $\varphi \mid\!\sim^{JZ}_{\{\varphi \Rightarrow \mathbf{F}\}} \mathbf{F}$, but not $\varphi \vdash \mathbf{F}$. However, it validates a similar requirement which seems more adequate for conditional approaches.

Theorem 7.3 (Consistency preservation)
Let $\Sigma \subset \mathcal{L}$, $\Delta \subseteq \mathcal{L}(\Rightarrow)$ be finite and σ be a conjunction of Σ's elements. Then $\Sigma \cup \Delta \mid\!\sim^{JZ} \mathbf{F}$ iff $\Delta \vdash^{\kappa\pi} \sigma \Rightarrow \mathbf{F}$

Another important type of inference principles, concerned with language dependence and irrelevance reasoning in the probabilistic context, has been discussed in [HK 95]. The corresponding invariance features for default inference turn out to be valid for system JZ as well.

Theorem 7.4 (Representation independence)
Let $\mathcal{L}', \mathcal{L}'' \subseteq \mathcal{L}$ be two sublanguages of propositional logic and $f : \mathcal{L}'(\Rightarrow) \rightarrow \mathcal{L}''(\Rightarrow)$ be a faithful embedding, i.e. for all $\varphi, \psi \in \mathcal{L}'$, $f(\varphi \wedge \psi) = f(\varphi) \wedge f(\psi)$, $f(\neg\varphi) = \neg f(\varphi)$, $f(\varphi \Rightarrow \psi) = f(\varphi) \Rightarrow f(\psi)$, and $\varphi \vdash \psi$ iff $f(\varphi) \vdash f(\psi)$. Let $f(X) = \{f(x) \mid x \in X\}$. Then

- *$\Sigma \cup \Delta \mid\!\sim^{JZ} \psi$ iff $f(\Sigma) \cup f(\Delta) \mid\!\sim^{JZ} f(\psi)$.*

Whereas the above result also holds for normality maximization entailment, this is no longer true for the next property, which deals with the handling of independent language fragments.

Theorem 7.5 (Minimum irrelevance)
Let $\mathcal{L}', \mathcal{L}'' \subseteq \mathcal{L}$ be two sublanguages of propositional logic with disjoint sets of propositional variables. Assume $\Sigma' \cup \{\psi\} \subseteq \mathcal{L}'$, $\Delta' \subseteq \mathcal{L}'(\Rightarrow)$, $\Sigma'' \subseteq \mathcal{L}''$, $\Delta'' \subseteq \mathcal{L}''(\Rightarrow)$, and $\Sigma'' \cup \Delta'' \not\mid\!\sim^{JZ} \mathbf{F}$. Then

- *$\Sigma' \cup \Sigma'' \cup \Delta' \cup \Delta'' \mid\!\sim^{JZ} \psi$ iff $\Sigma' \cup \Delta' \mid\!\sim^{JZ} \psi$.*

Recall that in probabilistic reasoning, entropy maximization satisfies minimum irrelevance, but fails to adher representation independence, which is seen by some authors as a serious drawback.

The intentions and construction techniques behind system JZ exhibit some obvious similarities with normality maximization, also documented by our denotation). But there is also a more substantial link between these approaches. In fact, for every finite default knowledge base Δ, there is a $\vdash^{\kappa\pi}$-equivalent Δ^* which produces the same conclusions for $\mid\!\sim^{JZ}$ as Δ for $\mid\!\sim^{NM}$.

Theorem 7.6 (Reconstruction)
Let $\Delta = \{\varphi_i \Rightarrow^{a_i} \psi_i \mid i \leq n\} \subseteq \mathcal{L}(\Rightarrow)$ and $B(\Delta)$ be the set of all boolean combinations of φ_i, ψ_i in normal form, which is finite. Set $\Delta^* = \{\varphi_i \Rightarrow^{a_i} \psi_i \vee \psi \mid i \leq n, \psi \in B(\Delta)\}$. Then $JZ[\Delta^*] = NM(R_0, \Delta)$ and $\mid\!\sim^{JZ}_{\Delta^*} = \mid\!\sim^{NM}_{\Delta}$.

This result also demonstrates that we cannot have left equivalence w.r.t. $\vdash^{\kappa\pi}$. But note that this opens the road for inheritance to exceptional subclasses.

- $\{\neg\varphi\} \cup \{\mathbf{T} \Rightarrow \varphi, \mathbf{T} \Rightarrow \psi\} \mid\!\sim^{JZ} \psi$, but
  $\{\neg\varphi\} \cup \{\mathbf{T} \Rightarrow \varphi, \mathbf{T} \Rightarrow (\varphi \leftrightarrow \psi)\} \not\mid\!\sim^{JZ} \psi$.

8 EXAMPLES

To get a feeling for how JZ-entailment works, we are going to take a closer look at some benchmark examples which we consider interesting for conceptual or technical reasons. For each example, we shall state the - presumably desirable - presence or absence of an inference pattern $\Sigma \cup \Delta \mid\!\sim^{JZ} \psi$, indicate the set of all shiftable parts $\mathcal{S}(\Delta) = \{\varphi \wedge \neg\psi \mid \varphi \Rightarrow \psi \in \Delta\}$, determine the approximation sequence $(R^i \mid i \leq s+1)$, identify the canonical JZ-model, and validate or invalidate $\Sigma \mid\!\sim^{JZ}_{\Delta} \psi$. To achieve better readability, we denote the $\mathcal{L}$-formulas by upper case letters A, B, P, Q. We also assume that A, B, P, Q are logically independent. In fact, w.l.o.g., we may interpret them as propositional variables. Because the following list is in no way meant to be exhaustive, the reader is invited to compute the JZ-construction for his or her favourite default reasoning pattern. We start with an instance of inheritance to exceptional subclasses.

1. Exceptional inheritance :

$\{A\} \cup \{A \Rightarrow P, P \Rightarrow Q, A \Rightarrow \neg Q, P \Rightarrow B\} \mid\!\sim B$,

$\mathcal{S}(\Delta) = \{A \wedge \neg P, A \wedge Q, P \wedge \neg Q, P \wedge \neg B\}$,

$R^1 = R_0[P \wedge \neg Q + 1][P \wedge \neg B + 1]$,

$R_0[P \wedge \neg Q + 1][P \wedge \neg B + 1][A \wedge \neg P + 2][A \wedge Q + 2]$,

$\mathrm{JZ}[\Delta](\neg B|A) = 1$, i.e. $\Sigma \cup \Delta \mid\!\sim^{JZ} B$.

The JZ-model is in fact the only justifiably constructible $\kappa\pi$-model of Δ.

2. Redundancy :

$\{\neg A, \neg P\} \cup \{\mathbf{T} \Rightarrow A, \neg A \Rightarrow P, \neg A \Rightarrow P \vee Q\} \not\mid\!\sim Q$,

$\mathcal{S}(\Delta) = \{\neg A, \neg A \wedge \neg P, \neg A \wedge \neg P \wedge \neg Q\}$,

$R^1 = R_0[\neg A + 1]$,

$R_0[\neg A + 1][\neg A \wedge \neg P + 1][\neg A \wedge \neg P \wedge \neg Q + 0]$,

$\mathrm{JZ}[\Delta](\neg Q|\neg A \wedge \neg P) = 0$, i.e. $\Sigma \cup \Delta \not\mid\!\sim^{JZ} Q$.

In the next example, we are only interested in the JZ-construction. Let $S = (A \vee B) \wedge (P \vee Q)$.

3. Nested crossing :

$\{\mathbf{T} \Rightarrow \neg(S \wedge A), \mathbf{T} \Rightarrow \neg(S \wedge B)$,

$\mathbf{T} \Rightarrow \neg(S \wedge P), \mathbf{T} \Rightarrow \neg(S \wedge Q)\}$,

$\mathrm{S}(\Delta) = \{S \wedge A, S \wedge B, S \wedge P, S \wedge Q\}$,

$R_0[S \wedge A + 1/2][S \wedge B + 1/2][S \wedge P + 1/2][S \wedge Q + 1/2]$,

The uniform shifting of the four shiftable propositions stops as soon as the threshold 1 is reached. Note that here we have an infinite number of justifiably constructible $\kappa\pi$-models. Next comes another blow against naive normality maximization.

4. Geffner's example :

$\{A, B, P\} \cup \{A \Rightarrow Q, P \Rightarrow Q, A \wedge B \Rightarrow \neg Q\} \not\mid\!\sim Q, \neg Q$,

$\mathcal{S}(\Delta) = \{A \wedge \neg Q, P \wedge \neg Q, A \wedge B \wedge Q\}$,

$R^1 = R_0[A \wedge \neg Q + 1][P \wedge \neg Q + 1]$,

$R_0[A \wedge \neg Q + 1][P \wedge \neg Q + 1][A \wedge B \wedge Q + 2]$,

$\mathrm{JZ}[\Delta](Q|A \wedge P \wedge B) = \mathrm{JZ}[\Delta](\neg Q|A \wedge P \wedge B) = 0$, i.e. $\Sigma \cup \Delta \not\mid\!\sim^{JZ} Q, \neg Q$.

Because $A \wedge B$ and P are unrelated, there is no reason to prefer either Q or $\neg Q$. The following example is very useful to exhibit the difference between quasi-probabilistic approaches with, and simple preference-based accounts without a proper independence notion.

5. Anti-prioritization :

$\{A \vee (P \wedge Q)\} \cup \{\mathbf{T} \Rightarrow \neg P, \mathbf{T} \Rightarrow \neg Q$,

$(A \vee P) \Rightarrow \neg A, (A \vee Q) \Rightarrow \neg A\} \not\mid\!\sim A, \neg A$,

$\mathcal{S}(\Delta) = \{P, Q, A\}$,

$R^1 = R_0[P + 1][Q + 1]$,

$R_0[P + 1][Q + 1][A + 2]$,

$\text{JZ}[\Delta](A|A \vee (P \wedge Q)) = \text{JZ}[\Delta](\neg A|A \vee (P \wedge Q)) = 0$,

i.e. $\Sigma \cup \Delta \not\mathrel{|\!\sim}^{JZ} A, \neg A$.

We shall come back to this example when discussing the relationship of system JZ with other default formalisms. The remaining inference pattern, forward chaining, holds for classical defeasible inheritance formalisms, but usually fails for quasi-probabilistic approaches, like system JZ, which allow backwards propagation of evidence.

6. Forward chaining :

$\{A\} \cup \{A \Rightarrow P, P \Rightarrow Q, Q \Rightarrow B, A \Rightarrow \neg B\} \mathrel{|\!\sim} Q, \neg B$.

However, it is possible to obtain a rule-based variant of system JZ which verifies this pattern.

9 JZ MEETS ME

Entropy maximization is a very well-behaved inference rule for choosing a most plausible estimate from a closed convex set of probability distributions. Together with its relativized version, cross entropy minimization, it is completely characterizable by some reasonable general desiderata for probabilistic inference policies [SJ 80, PV 90]. Additional support comes from concentration and large deviation theorems based on combinatorial and uniformity arguments [JAY 83,VCC 81,JAE 95].

To exploit this inference technique for default reasoning, we first have to translate the given default knowledge base $\Delta = \{\varphi_i \Rightarrow^{a_i} \psi_i \mid i \leq n\} \subseteq \mathcal{L}(\Rightarrow)$ into a suitable probabilistic notation. The main idea is to express the defaults $\varphi_i \Rightarrow^{a_i} \psi_i$ by conditional probability constraints of the form $P(\neg\psi_i|\varphi_i) \leq \varepsilon^{a_i}$, where ε is an arbitrary but fixed positive infinitesimal number, i.e. $0 < \varepsilon < \ldots < 1/3 < 1/2 < 1$. Alternatively, we may see ε as a parameter converging towards 0 [GMP 90]. Let $\Delta_\varepsilon^P = \{P(\neg\psi_i|\varphi_i) \leq \varepsilon^{a_i} \mid i \leq n\}$. Furthermore, for the sake of convenience, let $\mathcal{L}_0$ be a sublanguage of $\mathcal{L}$ restricted to finitely many variables with $\Delta \subseteq \mathcal{L}_0(\Rightarrow)$. But note that all our considerations are independent from the exact choice of $\mathcal{L}_0$. It is easy to see that Δ_ε^P defines a closed and convex set of (standard or nonstandard) probability distributions over $\mathcal{B}_{\mathcal{L}_0}$. These can be represented by finite tuples $\vec{x}$ with $\Sigma x_i = 1$ and $0 \leq x_i$. Entropy maximization means maximizing $-\Sigma x_i \log(x_i)$ under these additional constraints. Let $ME(\Delta_\varepsilon^P)$ be the entropy maximal model of Δ_ε^P. Then the ME-based default inference relation $\mathrel{|\!\sim}^{ME}$ can be defined as follows [GMP 90, WEY 95] (independently from $\mathcal{L}_0$).

Definition 9.1 (ME-inference)
Let $\Sigma \cup \{\psi\}$ be a finite subset of $\mathcal{L}_0$ and σ be a conjunction of Σ's elements.

$\Sigma \mathrel{|\!\sim}_\Delta^{ME} \psi$ *iff* $lim_{\varepsilon \to 0} ME(\Delta_\varepsilon^P)(\neg\psi|\sigma) = 0$,

$\Sigma \mathrel{|\!\sim}_\Delta^{ME} \psi$ *iff* $ME(\Delta_\varepsilon^P)(\neg\psi|\sigma)$ *is infinitesimal.*

On sufficiently simple - although unrealistic - default sets, it easily follows from the definitions and the results on sytem Z* that system JZ agrees with ME-inference.

Theorem 9.2 (Minimal core sets)
If Δ is a finite consistent minimal core default set, i.e. $\Delta^{\rightarrow}$ is logically irredundant, then $\mathrel{|\!\sim}_\Delta^{JZ} = \mathrel{|\!\sim}_\Delta^{ME}$.

In fact, the relationship is even closer than that. For the default knowledge bases Δ discussed above, we can show that the $\kappa\pi$-measure $R_{ME(\Delta_\varepsilon^P)}$ associated with the nontandard probabilistic ME-model $ME(\Delta_\varepsilon^P)$ is exactly the JZ-model of Δ. An interesting consequence of this observation is that κ-rankings - which pick up integer values - are not fine-grained enough to capture ME-inference on a semi-qualitative level (cf. nested crossing).

However, system JZ may also disagree with ME in some contexts, and it does so for good reasons. There are instances where the quantitative character of ME strongly conflicts with our qualitative intuitions about what should, or shouldn't affect the defeasible consequences of some given Δ. For instance, we may consider a variant of the nested crossing example where we replace P, Q by the four propositions P, Q, P', Q'. Although this doesn't seem to affect the qualitative character of the problem, it turns out that $R_{ME}(P \wedge Q|S)$ suddenly changes from 1/2 to 0, whereas $R_{JZ}(P \wedge Q|S)$ keeps the original value 1/2. This behaviour is even better illustrated by the following example.

- $\Delta = \{\mathbf{T} \Rightarrow A, \mathbf{T} \Rightarrow B, \mathbf{T} \Rightarrow A \wedge B\}$.

Then $\text{JZ}[\Delta] = R_0[\neg A + 1/2][\neg B + 1/2][\neg A \vee \neg B + 1/2]$, whereas $\text{ME}[\Delta] = R_0[\neg A \vee \neg B + 1]$. Therefore, we have $\{\neg A\} \cup \Delta \not\mathrel{|\!\sim}^{ME} B$ but $\{\neg A\} \cup \Delta \mathrel{|\!\sim}^{JZ} B$.

A major problem with $\mathrel{|\!\sim}^{ME}$ is its sensitivity to small changes - within the same order of magnitude - of the conditional probability bounds. For instance, given $0 < r_i$, let $\Delta_{\vec{r}\varepsilon}^P = \{P(\neg\psi_i|\varphi_i) \leq r_i\varepsilon^{a_i} \mid i \leq n\}$. Then, $ME(\Delta_\varepsilon^P)$ and $ME(\Delta_{\vec{r}\varepsilon}^P)$ may support completely different conclusions (cf. nested crossing). In addition, the lack of representation independence of ME also affects the corresponding default inference notion $\mathrel{|\!\sim}^{ME}$.

10 COMPARISONS

There are many competing default formalisms based on alternative but similarly sophisticated ranking or prioritization procedures. An exact comparison would require lengthy technical considerations, which are beyond the scope of the present paper. So, we are going to restrict ourselves to some general remarks and illustrative examples. We begin with a look at the comparative strength of the epistemic construction entailment relations. Let $\mid\!\sim^{P}$ be standard preferential entailment [KLM 90]. Recall that $\mid\!\sim^{NM}$ is equivalent to system Z. Obviously, we have $\mid\!\sim^{P} \subset \mid\!\sim^{J} \subset \mid\!\sim^{JJ} \subset \mid\!\sim^{JZ}, \mid\!\sim^{JJR}$. $\mid\!\sim^{J}, \mid\!\sim^{JJ}, \mid\!\sim^{JZ}$ and $\mid\!\sim^{JJR}$ are all incomparable with $\mid\!\sim^{NM}$.

JZ-entailment itself is orthogonal to $\mid\!\sim^{JJR}$ but also to prioritized accounts, like conditional entailment [GP 92], default counting proposals, like lexicographic closure [BCDLP 93], and strong belief function approaches, like LCD-inference [BSS 95]. Whereas all these default formalisms are preferential consequence relations, only lexicographic closure verifies rational monotony. A very interesting competitor is LCD-entailment. Not only does it validate the usual inference desiderata, it also has - although rather different in practice - a similar flavour as system JZ. It was originally developed in the context of belief function theory, but can be easily translated into our framework.

Nevertheless, all the traditional prioritized accounts are haunted by their ad hoc handling of priorities, which doesn't take into account implicit independencies in a systematic way. Furthermore, the computation of the overall hierarchical structure is usually done at the beginning, there is no dynamic incremental prioritization as for system JZ. The Anti-prioritization example illustrates well the problems of many prioritized approaches.

We know that the JZ-model is $R_{JZ} = R_0[P+1][Q+1][A+2]$. It follows that $R_{JZ}(A|A \vee (P \wedge Q)) = R_{JZ}(\neg A|A \vee (P \wedge Q)) = 0$. This seems reasonable because A and $P \wedge Q$ are - in a basically equivalent way - second degree exceptional. Conditional entailment and lexicographic closure, on the other hand, are showing a more asymmetric behaviour. They always prefer the violation of several defaults of lower priority to the falsification of any default of higher priority. Consequently, both will defeasibly infer $\neg A$.

Although the LCD-approach also stays ambiguous in this example, it does so in a stronger, different way. Whereas system JZ requires $R_{JZ}(A) = R_{JZ}(P \wedge Q)$, LCD doesn't seem to make any statements about the relative positions of A and $P \wedge Q$. Now suppose, we add a further default $(B \vee (P \wedge Q)) \Rightarrow \neg B$. Then $R_{JZ}(A) < R_{JZ}(B)$, which seems plausible. But this preference fails to hold for LCD. One reason for its problems is the different handling of single - as A or B - and conjoined - as $P \wedge Q$ - exceptional propositions.

Of considerable interest is also coherence closure [TP 95], a simple ranking-based proposal. It implements normality maximization together with the requirement that worlds which violate less defaults than other ones - w.r.t. $\subset$ - automatically should get a lower rank. It follows that this approach handles the redundancy example in a different way than system JZ. Furthermore, it also doesn't satisfy - mutans mutandum - the epistemic construction principle. In fact, coherence closure cannot deal with independencies in a probabilistically coherent way. The reason is that it doesn't sufficiently take into account the strength of the violated defaults.

11 CONCLUSIONS

System JZ is a new quasi-probabilistic default inference notion based on the uniform minimal iterated update construction of a canonical $\kappa\pi$-model from any graded default knowledge base. It satisfies all of Lehmann's rationality postulates, Halpern and Koller's invariance principles, as well as the usual default reasoning patterns, in particular inheritance to exceptional subclasses.

System JZ seems particularly appropriate if we are looking for a canonical model, e.g. for decision-taking, if we are interested in the automatic extraction of implicit independencies from a default knowledge base, and if we want to detect and drop redundant information. System JZ is also closely linked to ME-based default inference, but easier to handle and less sensitive to quantitative effects. Although it doesn't fully share neither the simplicity nor the computational features of system Z, it still constitutes a successful implementation of its construction philosophy in the context of order-of-magnitude probabilities and the minimal information philosophy.

REFERENCES

BCDLP 93 S. Benferhat, C. Cayrol, D. Dubois, J. Lang, H. Prade. Inconsistency management and prioritized syntax-based entailment. In *Proceedings of IJCAI 93*, Morgan Kaufmann, 1993.

BDP 94 S. Benferhat, C. D. Dubois, H. Prade. Expressing independence in a possibilistic framework and its application to default reasoning. In *Proceedings of ECAI 94*, Amsterdam, 1994.

BGKH 97 F. Bacchus, A.J. Grove, D. Koller, J. Y. Halpern. Statistical foundations for default reasoning. In *Artificial Intelligence*, 87: 75-143, 1997.

BSS 95 S. Benferhat, A. Saffiotti, P. Smets. Belief functions and default reasoning. In *Proceedings of UAI 95.* Morgan Kaufmann, 1995.

DB 88 D. Dubois, H. Prade. *Possibility Theory.* Plenum Press, New York 1988.

GP 92 H. Geffner, J. Pearl. Conditional entailment : bridging two approaches to default reasoning. In *Artificial Intelligence*, 53: 209 - 244, 1992.

GOP 96 M. Goldszmidt, J. Pearl. Qualitative probabilities for default reasoning, belief revision, and causal modeling. In *Artificial Intelligence*, 84 : 57-112, 1996.

GMP 90 M. Goldszmidt, P. Morris, J. Pearl. A maximum entropy approach to nonmonotonic reasoning. In *Proceedings of AAAI 90.* Morgan Kaufmann 1990.

HK 95 J. Halpern and D. Koller. Representation dependence in probabilistic inference. In *Proceedings of IJCAI 95*, Morgan Kaufmann, 1995.

JAY 83 E.T. Jaynes. Where do we stand on maximum entropy ? In R.D. Rosenkrantz (ed.) E.T. Jaynes : *Papers on Probability, Statistics and Statistical Physics.* D. Reidel Publishing, Boston, 1983.

JAE 95 M. Jaeger. *Default Reasoning about Probabilities.* PhD Thesis, Saarbuecken 1995.

KLM 90 S. Kraus, D. Lehmann, M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. In *Artificial Intelligence*, 44: 167-207, 1990.

LM 92 D. Lehmann, M. Magidor. What does a conditional knowledge base entail ? In *Artificial Intelligence*, 55:1-60, 1992.

PEA 89 J. Pearl. Probabilistic semantics for nonmonotonic reasoning. A survey. In *Proceedings of KR 89.* Morgan Kaufmann. Toronto, 1989.

PEA 90 J. Pearl. System Z: a natural ordering of defaults with tractable applications to nonmonotonic reasoning. In *Proceedings of TARK 90.* Morgan Kaufmann, 1990.

PV 90 J.B. Paris and A. Vencovska. Anote on the inevitability of maximum entropy.In *Internationl Journal of Approximate Reasoning*, 4:183-223, 1990.

SJ 80 J.E. Shore and R.W. Johnson. Axiomatic derivation of the principle of cross-entropy minimization. In *IEEE Transactions on Information Theory*, IT-26(1):26-37, 1980.

SPO 88 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.* D. Reidel, Dordrecht, Netherlands, 1988.

SPO 90 W. Spohn. A general non-probabilistic theory of inductive reasoning. In R.D.Shachter et al. (eds.) *Uncertainty in Artificial Intelligence 4*, North-Holland, Amsterdam 1990.

TP 95 SW. Tan and J. Pearl. Specificity and inheritance in default reasoning. In Proceedings of IJ-CAI 95. Morgan Kaufmann, 1995.

VCC 81 J.M. van Campenhout and T:M: Cover. Maximum entropy and conditional probability. In *IEEE Transactions on Information Theory*, IT-27(4):483-489, 1981.

WEY 91 E. Weydert. Qualitative magnitude reasoning. Towards a new semantics for default reasoning. In J. Dix et al. (eds.), *Nonmonotonic and Inductive Reasoning.* Springer-Verlag 1991.

WEY 93 E. Weydert. Plausible inference for default conditionals. In *Proceedings of ECSQARU 93.* Springer, Berlin, 1993.

WEY 94 E. Weydert. General belief measures. In *Proceedings of UAI 94.* Morgan Kaufmann, 1994.

WEY 95 E. Weydert. Defaults and infinitesimals. Defeasible inference by non-archimdean entropy maximization. In *Proceedings of UAI 95.* Morgan Kaufmann, 1995.

WEY 96 E. Weydert. System J - revision entailment. In *Proceedings of FAPR 96*, Springer-Verlag, 1996.

WEY 98 E. Weydert. Ranking constructions and entropy maximization. Submitted, 1998.

Pointwise Circumscription Revisited

Eyal Amir
Department of Computer Science,
Gates Building, 2A wing
Stanford University, Stanford, CA 94305-9020, USA
eyala@cs.stanford.edu

Abstract

A decade ago, Pointwise Circumscription was proposed as a tool for formalizing common sense. In this paper, we revisit some of its uses and examine some cases in which it does not yield the intended behavior. Specifically, we explore the cases in which unsatisfiability may result from the presence of multiple minimal models (for McCarthy's Circumscription). Then, we present a form of circumscription, called *Point-Sensitive Circumscription*, that is a generalization of McCarthy's Circumscription and Lifschitz's Generalized Pointwise Circumscription. We illustrate how Point-Sensitive Circumscription handles these cases without losing the control over selective fine-grained variance of predicates and functions. Last, we compare the two tools and their potential uses in formalizing Theories of Action.

1 INTRODUCTION

Pointwise Circumscription, devised by Lifschitz in (Lifschitz, 1987) and (Lifschitz, 1989), is a nonmonotonic method defined along the intuitions of Circumscription (McCarthy, 1986) with superior control of the minimization process. Since its debut, it has been used in formalizing Common Sense problems and solutions, such as in (Doherty and Lukaszewicz, 1994), formalizing some Entailment Classes in the theory of *Features and Fluents* (cf (Sandewall, 1994)), and in (Amir, 1997), using Pointwise Circumscription to formalize the system of (Lin and Reiter, 1994). It was argued in (Lifschitz, 1987) and (Doherty and Lukaszewicz, 1994) that Pointwise Circumscription has the power to be a tool for formalizing solutions for the Frame Problem.

In this paper, we show that there are cases in which Pointwise Circumscription requires the intended model to be a minimum, rather than merely minimal (sections 2,3). We present a form of circumscription, called *Point-Sensitive Circumscription*, that is a generalization of McCarthy's Circumscription and Lifschitz's Generalized Pointwise Circumscription (sections 4,5.1). Point-Sensitive Circumscription allows us to get a consistent theory where Pointwise Circumscription fails to do so, while maintaining similar control over the minimization process. For Point-Sensitive Circumscription, we allow a varied range of point-wise-ness (the removal of elements of the minimized predicate separately), but maintain the same control over selective fine-grained variance of predicates and functions. We give semantics for our approach and demonstrate the different behavior using examples from the literature (section 5).

In this paper, we use the following two conventions: (1) the relation $p < q$ between two propositions refers to the usual ordering of Boolean values ($false < true$), with $p \leq q$ allowing $p = q$; and (2) the relation $P < Q$ between two predicates with same arity refers to the strict subset relation (regarding P, Q as the relative sets of elements).

2 POINTWISE CIRCUMSCRIPTION

2.1 CIRCUMSCRIPTION

McCarthy's Circumscription (McCarthy, 1980) is one of the first and major nonmonotonic reasoning tools. Since its debut, the nonmonotonic reasoning line of work has expanded and several textbooks now exist that give a fair view of nonmonotonic reasoning and its uses (e.g., (Brewka, 1991), (Antoniou, 1997), (Brewka et al., 1997), (D.M. Gabbay, 1994), (Sandewall, 1994), (Shanahan, 1997)). The motivations for nonmonotonic reasoning vary from formalizing Common Sense reasoning through Elaboration Tolerance and representation of uncertainty to Belief Revision. We do not expand on these motivations further here, but section 5 gives an example of the Common Sense application, and the reader may look at (Shanahan, 1997),(McCarthy, 1998),(Pearl, 1990) and (Antoniou, 1997) for further details in these directions.

McCarthy's Circumscription formula (McCarthy, 1980)

$$\begin{aligned}&Circ[A(P,Z);P;Z] =\\ &\quad A(P,Z) \wedge \forall p,z\,(A(p,z) \implies \neg(p < P))\end{aligned}$$

says that in the theory A, with parameter relations and function sequences P, Z, P is a minimal element such that $A(P,Z)$ is still consistent, when we are allowed to vary Z in order to allow P to become smaller.

Take for example the following simple theory:

$$T \equiv block(B_1) \wedge block(B_2)$$

Then, the circumscription of *block* in T, varying nothing, is $Circ[T; block;] = T \wedge \forall p\,[T_{[block/p]} \implies \neg(p < block)]$ which is equivalent to

$$Circ[T; block;] \equiv \forall x\,(block(x) \Leftrightarrow (x = B_1 \vee x = B_2))$$

By minimizing *block* we concluded that there are no other blocks in the world other than those mentioned in the original theory T.

2.2 POINTWISE CIRCUMSCRIPTION: THE FORMULA

Pointwise Circumscription was first proposed in (Lifschitz, 1986) and then expanded in (Lifschitz, 1987) and (Lifschitz, 1989). In (Doherty and Lukaszewicz, 1994) it was demonstrated that Pointwise Circumscription can be used to formalize various logics of action[1] in the framework of Features and Fluents (Sandewall, 1994). These logics include OCM (Original Chronological Minimization), PCM (Prototypical Chronological Minimization), PCMF (PCM with Filtering), CMON (Chronological Minimization of Occlusion with Nochange Premises) and CMOC (Chronological Minimization of Occlusion and Change).

The general idea behind Pointwise Circumscription is to capture the minimization idea of *Circumscription* (McCarthy, 1980), (McCarthy, 1986), while making the circumscription on different subsets of the domain for different stages of the minimization process.

The basic case for Pointwise Circumscription is the formula

$$A(P) \wedge \forall x \neg[Px \wedge A(\lambda y(Py \wedge x \neq y))]$$

(where we minimize the subscript P again, varying nothing), intuitively saying that we remove one element at a time. One of the benefits of such an approach is the first-orderedness of the circumscription formula. This property disappears, though, in the most general form, which we use below.

[1]Some of the formalizations needed *filtering*.

For Pointwise Circumscription, let $A(S_1, ..., S_n)$ be a sentence in which each S_i is a predicate symbol or a function symbol (in particular, it can be a 0-ary function symbol, i.e., an object constant). We want to minimize one of the predicate symbols from this list, S_{i_0} (Thus, S_{i_0} corresponds to P, and the other members of the list correspond to Z in the notation used above). Let us write $EQ_V(P,Q)$ ("P and Q are equal outside V") for

$$EQ_V(P,Q) \overset{def}{=} \forall x(\neg Vx \implies (Px \equiv Qx)) \qquad (1)$$

for P, Q predicates or functions, and V a predicate, all with same arity. The *Pointwise circumscription* of S_{i_0} in A with S_i allowed to vary on V_i is, by definition,

$$A(S) \wedge \forall xs \neg[S_{i_0}x \wedge \neg s_{i_0}x \wedge \bigwedge_{i=1}^{n} EQ_{V_i x}(s_i, S_i) \wedge A(s)] \qquad (2)$$

Here, S stands for $S_1, ..., S_n$, s is a list $s_1, ..., s_n$ of predicate and function variables corresponding to the predicate and function constants S, and $\lambda xuV_i(x,u)$ $(i = 1, ..., n)$ is a predicate without parameters that does not contain $S_1, ..., S_n$ and whose arity is the arity of S_{i_0} plus the arity of S_i. We denote (2) by $C_{PW}[A; S_{i_0}; S_1/V_1, ..., S_n/V_n]$.

2.2.1 An Intuitive Explanation of Pointwise Circumscription

The Pointwise Circumscription formula (2) has the property that we remove x from S_{i_0} while varying S (recall that S is the list of predicates/functions $S_1, ..., S_n$) in each of the $V_i x$ separately, fixing all the other parts of the domain, at that time.

In other words, the bracketed part of the formula says that, in the process of removing x from S_{i_0}, we allow only some parts of S to change. These parts are determined by $V_i x$ by saying that only where $V_i x$ is TRUE s_i may be different from S_i (recall that $EQ_{V_i x}(s_i, S_i)$ means that s_i is equal to S_i outside $\lambda u V_i(x,u)$).

2.3 SEMANTICS FOR POINTWISE CIRCUMSCRIPTION

In (Lifschitz, 1987), Pointwise Circumscription was given semantic interpretation.

Take, without loss of generality, $i_0 = 1$. For a model $\mathcal{M}$ of $A(S)$, let $|\mathcal{M}|$ be the associated universe, and for every term, function or predicate a, $a^{\mathcal{M}}$ is the realization of a in $\mathcal{M}$.

Definition 2.1 *Let $\mathcal{M}_1, \mathcal{M}_2$ have the same universe U, and let $\xi \in U^k$, where k is the arity of S_1. We say that $\mathcal{M}_1 \leq^{\xi} \mathcal{M}_2$ iff*

1. *$K^{\mathcal{M}_1} = K^{\mathcal{M}_2}$ for every function or predicate constant K that is not in S,*
2. *for any $i = 1, ..., n$, $S_i^{\mathcal{M}_1}$ and $S_i^{\mathcal{M}_2}$ coincide on $\{\eta \mid \neg V_i^{\mathcal{M}_1}(\xi, \eta)\}$*
3. *$S_1^{\mathcal{M}_1}(\xi) \implies S_1^{\mathcal{M}_2}(\xi)$.*

Proposition 2.2 ((Lifschitz, 1987)) *Let $\mathcal{M}$ be a model of $A(S)$.*

$$\begin{aligned}\mathcal{M} \models C_{PW}[A; S_1; S_1/V_1, ..., S_n/V_n] \iff \\ \forall \mathcal{M}' \in [A(S)]\ \forall \xi \in |\mathcal{M}|^k \\ \neg(\mathcal{M}' \leq^{\xi} \mathcal{M} \wedge \mathcal{M} \not\leq^{\xi} \mathcal{M}')\end{aligned}$$

Other words, $\mathcal{M} \models C_{PW}[A; S_1; S_1/V_1, ..., S_n/V_n]$ iff for each $\xi \in |\mathcal{M}|^k$, $\mathcal{M}$ is minimal relative to $\leq^{\xi}$.

3 A COUNTER-INTUITIVE EXAMPLE

In this section, we discuss only one simple example.

Let $A(P) = (P(X) \vee P(Y)) \wedge X \neq Y$. For this simple language (the only predicate is P, and there are exactly two object constant symbols, X, Y), we get the Pointwise Circumscription formula

$$A(P) \wedge \forall x p \neg [Px \wedge \neg px \wedge EQ_{V_{Px}}(p, P) \wedge A(p)].$$

Assume that for all x, $(V_P x) \equiv True$, so that p is allowed to differ from P with no limitations. We get the further simplified Pointwise Circumscription formula

$$A(P) \wedge \forall x p \neg [Px \wedge \neg px \wedge A(p)].$$

Let $C = C_{PW}[(P(X) \vee P(Y)) \wedge X \neq Y; P; P/\lambda x.True]$.

Proposition 3.1 $C \models FALSE$.

Proof

$$\begin{aligned}
& C_{PW}[A(P); P; P/\lambda x.True] \equiv \\
& A(P) \wedge \forall x p \neg [P(x) \wedge \neg p(x) \wedge A(p)] \equiv \\
& A(P) \wedge \\
& \quad \forall x p \neg [P(x) \wedge \neg p(x) \wedge (p(X) \vee p(Y)) \wedge X \neq Y] \\
& \implies \\
& A(P) \wedge \\
& \quad \forall p \neg \left[\begin{array}{l} P(X) \wedge \neg p(X) \wedge (p(X) \vee p(Y)) \wedge X \neq Y \vee \\ P(Y) \wedge \neg p(Y) \wedge (p(X) \vee p(Y)) \wedge X \neq Y \end{array} \right] \\
& \equiv \\
& A(P) \wedge \forall p \neg \left[\begin{array}{l} P(X) \wedge \neg p(X) \wedge p(Y) \wedge X \neq Y \vee \\ P(Y) \wedge \neg p(Y) \wedge p(X) \wedge X \neq Y \end{array} \right]
\end{aligned}$$

Let $p_x = \{X\}, p_y = \{Y\}$ be two possible values of p. Then, from the formula on the last line, we get

$$\begin{aligned}
& A(P) \wedge \forall p \neg \left[\begin{array}{l} P(X) \wedge \neg p(X) \wedge p(Y) \wedge X \neq Y \vee \\ P(Y) \wedge \neg p(Y) \wedge p(X) \wedge X \neq Y \end{array} \right] \\
& \implies \\
& A(P) \wedge \neg \left[\begin{array}{l} P(X) \wedge \neg p_x(X) \wedge p_x(Y) \wedge X \neq Y \vee \\ P(Y) \wedge \neg p_x(Y) \wedge p_x(X) \wedge X \neq Y \vee \\ P(X) \wedge \neg p_y(X) \wedge p_y(Y) \wedge X \neq Y \vee \\ P(Y) \wedge \neg p_y(Y) \wedge p_y(X) \wedge X \neq Y \end{array} \right] \\
& \equiv \\
& A(P) \wedge \\
& \quad \neg \left[\begin{array}{l} FALSE \vee \\ P(Y) \wedge (TRUE \iff X \neq Y) \wedge X \neq Y \vee \\ P(X) \wedge (TRUE \iff X \neq Y) \wedge X \neq Y \vee \\ FALSE \end{array} \right] \\
& \equiv \\
& A(P) \wedge \neg [(P(Y) \vee P(X)) \wedge X \neq Y] \equiv \\
& P(X) \vee P(Y) \wedge X \neq Y \wedge \\
& \quad \neg [(P(Y) \vee P(X)) \wedge X \neq Y] \equiv \\
& FALSE
\end{aligned}$$

■

This proposition reveals a limitation of Pointwise Circumscription, i.e., the requirement, in some cases, that P be a *minimum* rather than *minimal*[2].

This conclusion is supported by the semantics given by Lifschitz (proposition 2.2 above). Let $U = \{x, y\}$ be the set of elements in the universe. Let M_X, M_Y, M_{XY}, $M_\emptyset$ be the models with universe U, with X, Y interpreted to x, y, respectively, and the following interpretations for the predicate P:

$$P^{M_X} = \{X\} \qquad P^{M_Y} = \{Y\}$$
$$P^{M_{XY}} = \{XY\} \quad P^{M_\emptyset} = \emptyset$$

Figure 1 below displays the different models for $A(P)$.

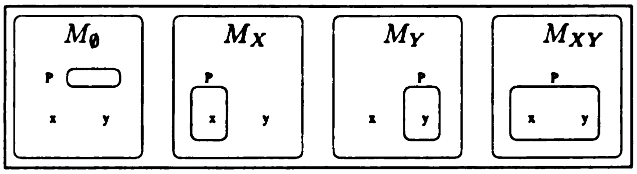


Figure 1: The four models of $A(P)$ with universe $U = \{x, y\}$.

There are exactly two elements in U, and thus there are two partial orders $\leq^x, \leq^y$ between the models. In order to check these partial orders, we first notice that the first two

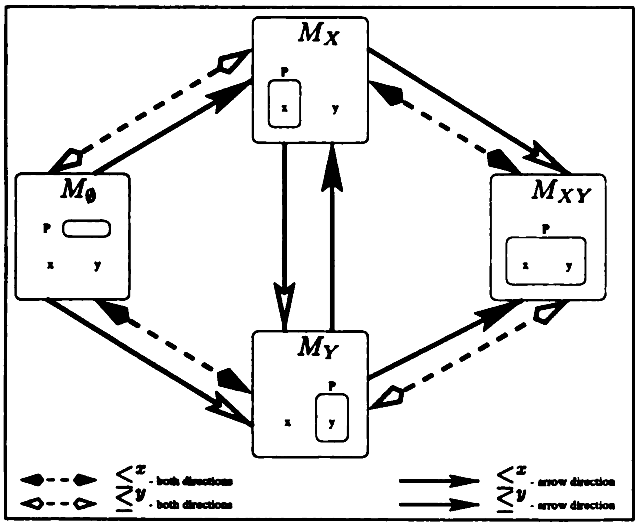
[2]Here *minimum* refers to an object that is smaller (in terms of the given partial order relation) than all other objects, while *minimal* refers to objects for which there is no smaller object.

conditions of definition 2.1 are met by any pair of models from the four examined above. Using the third condition, we get the obvious $M_\emptyset \leq^y M_X \leq^y M_{XY}$, $M_{XY} \not\leq^y M_X$, $M_\emptyset \leq^x M_Y \leq^x M_{XY}$ and $M_{XY} \not\leq^x M_Y$. We also get the somewhat less obvious $M_\emptyset \leq^x M_Y \leq^x M_X$, $M_X \not\leq^x M_Y$, $M_\emptyset \leq^y M_X \leq^y M_Y$ and $M_Y \not\leq^y M_X$.

Examining the conditions of proposition 2.2 for each one of the models, we reveal the following:

- $M_\emptyset \not\models C$ because $M_\emptyset \not\models A(P)$.
- $M_X \not\models C$ because $M_Y \models A(P)$ and $M_Y \leq^x M_X$ and $M_X \not\leq^x M_Y$.
- $M_Y \not\models C$ because $M_X \models A(P)$ and $M_X \leq^y M_Y$ and $M_Y \not\leq^y M_X$.
- $M_{XY} \not\models C$ because $M_X \models A(P)$ and $M_X \leq^x M_{XY}$ and $M_{XY} \not\leq^y M_X$.

Thus, again, there is no model that satisfies C. Figure 2 below displays the two orders on models.



M_X
$M_\emptyset$
M_{XY}
M_Y
P
x
y
$\leq^x$ - both directions
$\leq^y$ - both directions
$\leq^x$ - arrow direction
$\leq^y$ - arrow direction

Figure 2: The orders $\leq^x$ and $\leq^y$. The different colors represent the different orders.

4 POINT-SENSITIVE CIRCUMSCRIPTION

4.1 THE FORMULA

We present *Point-Sensitive Circumscription*, a modified version of Pointwise Circumscription in which the minimized predicate is minimized according to a *minimization domain*. This minimization domain may be a point and may be the complete set of elements. We preserve the ability to *select/vary* parts of the theory/domain dynamically.

It is important to notice at this point that Lifschitz already proposed a global circumscription that has some control over the varied domain (see formula (14) in (Lifschitz, 1987)). We shall compare that proposal with Pointwise Circumscription and Point-Sensitive Circumscription in section 5.

We use similar notations to those used in section 2. Let $A(S_1, ..., S_n)$ be a sentence in which each S_i is a predicate symbol or a function symbol (in particular, it can be a 0-ary function symbol, i.e., an object constant). We want to minimize one of the *predicate* symbols from this list, S_{i_0}.

In addition to the definition of EQ_V in (1), let us write $LS_R(P, Q)$ ("P is smaller than Q in the region R", or $P \cap R \subsetneq Q \cap R$) for

$$LS_R(P,Q) \stackrel{def}{\equiv} \forall x(Rx \Rightarrow (Px \Rightarrow Qx)) \wedge \exists x(Rx \wedge \neg Px \wedge Qx) \quad (3)$$

for P, Q predicates or functions, and R a predicate, all with same arity. The simplest example is where R is uniformly true, in which case $LS_{Rx}(P, Q) \equiv P \subsetneq Q$.

The *Point-Sensitive Circumscription* of S_{i_0} in A with S_i allowed to vary on V_i, and S_{i_0} minimized using R, denoted $C_{PS}[A; S_{i_0}/R; S_1/V_1, ..., S_n/V_n]$, is, by definition,

$$A(S) \wedge \forall xs \neg[LS_{Rx}(s_{i_0}, S_{i_0}) \wedge \bigwedge_{i=1}^{n} EQ_{V_ix}(s_i, S_i) \wedge A(s)] \quad (4)$$

where S stands for $S_1, ..., S_n$, s is a list $s_1, ..., s_n$ of predicate and function variables corresponding to the predicate and function constants S, and $\lambda xuR(x, u)$, $\lambda xuV_i(x, u)$ ($i = 1, ..., n$) are predicates without parameters that do not contain $S_1, ..., S_n$ and whose arity is the arity of S_{i_0} plus the arity of S_{i_0} and S_i, respectively (here $Rx = \lambda uR(x, u)$, $V_ix = \lambda uV_i(x, u)$).

4.1.1 An Intuitive Explanation

The bracketed part of the Point-Sensitive Circumscription formula (4) says that, in the process of making S_{i_0} smaller, we use R to define what "smaller" means and we allow only some parts of S to change. These parts are determined by Rx and V_ix by saying that s_i may be different from S_i only where V_ix is TRUE (recall that $EQ_{V_ix}(s_i, S_i)$ means that s_i is equal to S_i outside $\lambda u(V_i(x, u))$) and that we actually care if s_{i_0} is smaller only where Rx is TRUE.

Notice that the only difference between the Point-Sensitive Circumscription formula (4) and the Pointwise Circumscription formula (2) is in the first component of the bracketed formula. In that part, we changed the term $S_{i_0}x \wedge \neg s_{i_0}x$ (in (2)) to $LS_{Rx}(s_{i_0}, S_{i_0})$ (in (4)).

It is also important to notice that even for the cases where R is uniformly TRUE (in these cases, x is used only in the second component in the formula in parentheses), the second component is actually a conjunction of formulas, all using x. Therefore, although we do not require that x be removed from the predicate S_{i_0}, we do require that the variance of the different predicates/functions (including S_{i_0}) be correlated with x as an index.

4.2 SEMANTICS FOR POINT-SENSITIVE CIRCUMSCRIPTION

We follow the lines of (Lifschitz, 1987) with regard to the semantics for our Point-Sensitive Circumscription.

Take, without loss of generality, $i_0 = 1$. For a model $\mathcal{M}$ of $A(S)$, let $|\mathcal{M}|$ be the associated universe and, for every term, function or predicate a, $a^{\mathcal{M}}$ is the realization of a in $\mathcal{M}$. We write the corresponding definition to 2.1.

Definition 4.1 *Let $\mathcal{M}_1, \mathcal{M}_2$ have the same universe U, and let $\xi \in U^k$, where k is the arity of S_1. We say that $\mathcal{M}_1 \ll^{\xi} \mathcal{M}_2$ (a strict partial order) iff*

1. *$K^{\mathcal{M}_1} = K^{\mathcal{M}_2}$ for every function or predicate constant K that is not in S,*
2. *for any $i = 1, ..., n$, $S_i^{\mathcal{M}_1}$ and $S_i^{\mathcal{M}_2}$ coincide on $\{\eta \mid \neg V_i^{\mathcal{M}_1}(\xi, \eta)\}$*
3. *$LS_{R(\xi)}(S_1^{\mathcal{M}_1}, S_1^{\mathcal{M}_2})$ $(R(\xi) = \lambda u R(\xi, u))$.*

Let $[A(S)]$ be the set of models of $A(S)$. The following proposition says that every model of the circumscription formula for $A(S)$ is minimal in $[A(S)]$ according to all of the orders $\ll^{\xi}$, and vice versa.

Proposition 4.2 (Semantics) *Let $\mathcal{M} \in [A(S)]$.*

$$\mathcal{M} \models C_{PS}[A; S_1/R; S_1/V_1, ..., S_n/V_n] \iff \\ \forall \mathcal{M}' \in [A(S)]\ \forall \xi \in |\mathcal{M}|^k\ \neg(\mathcal{M}' \ll^{\xi} \mathcal{M})$$

PROOF For the forward direction, let us take $\mathcal{M} \models C_{PS}[A; S_1/R; S_1/V_1, ..., S_n/V_n]$. Assume that $\mathcal{M}' \in [A(S)]$ $\mathcal{M}' \ll^{\xi} \mathcal{M}$ for some $\xi \in |\mathcal{M}|^k$. Then $LS_{R(\xi)}(S_1^{\mathcal{M}'}, S_1^{\mathcal{M}})$, by definition 4.1 (requirement 3).

Take s to be $S^{\mathcal{M}'}$, the sequence of predicates/functions interpreting S (the sequence of predicate/function constant symbols) in $\mathcal{M}'$. For $x = \xi$ (ξ picked above), we get that the following are true in the model $\mathcal{M}$:

- $LS_{R(\xi)}(s_1, S_1)$, because $s = (s_1, ..., s_n)$ (that is our notation), and the way we picked ξ (s.t., $s_1 = S_1^{\mathcal{M}'}$ and $LS_{R(\xi)}(S_1^{\mathcal{M}'}, S_1^{\mathcal{M}})$),
- $A(s)$, because $s = S^{\mathcal{M}'}$ and $M' \models A(S)$, and
- $\bigwedge_{i=1}^{n} EQ_{V_i x}(s_i, S_i)$, because of definition 4.1 (requirement 2).

This contradicts $\mathcal{M} \models C_{PS}[A; S_1/R; S_1/V_1, ..., S_n/V_n]$, because we found x and s that satisfy the bracketed part of formula (4).

The reverse direction works by the same method. If $\mathcal{M}$ is $\ll^{\xi}$-minimal for every $\xi \in |\mathcal{M}|^k$, but does not satisfy formula (4), then there are s, x such that $LS_{R(\xi)}(s_1, S_1) \wedge \bigwedge_{i=1}^{n} EQ_{V_i x}(s_i, S_i) \wedge A(s)$. But then we can build a model $\mathcal{M}'$ with the same universe, taking all the constants other than those in S to be the same, and the constants of S take the values of s.

For $\mathcal{M}'$, we show now that $\mathcal{M}' \ll^{x} \mathcal{M}$. We follow the conditions of definition 4.1 one by one.

1. is satisfied by the construction of $\mathcal{M}'$.
2. is satisfied because s satisfies the second component of the bracketed part of the C_{PS} formula.
3. is satisfied because s satisfies the first component of the bracketed part of the C_{PS} formula.

We have found a model $\mathcal{M}'$ with $\mathcal{M}' \ll^{x} \mathcal{M}$. Contradiction. ■

Lemma 4.3 *For $\xi \in |\mathcal{M}|^k$, $\ll^{\xi}$ is a strict partial order over models of $\mathcal{L}$.*

PROOF Irreflexivity is simply because $LS_R(P, Q)$ is irreflexive ($R \cap P \subsetneq Q \cap R$). It is now enough to prove transitivity. Let $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3$ with same domain, and $\xi \in |\mathcal{M}_1|^k$. Assume $\mathcal{M}_1 \leq^{\xi} \mathcal{M}_2$ and $\mathcal{M}_2 \leq^{\xi} \mathcal{M}_3$. We prove the conditions for $\mathcal{M}_1 \leq^{\xi} \mathcal{M}_3$ one by one.

1. For K not in S, $K^{\mathcal{M}_1} = K^{\mathcal{M}_2} = K^{\mathcal{M}_3}$.
2. For every $i \in \{1, ..., n\}$, $S_i^{\mathcal{M}_1}, S_i^{\mathcal{M}_2}$ coincide on $\{\eta \mid \neg V_i^{\mathcal{M}_1}(\xi, \eta)\}$, and $S_i^{\mathcal{M}_2}, S_i^{\mathcal{M}_3}$ coincide on $\{\eta \mid \neg V_i^{\mathcal{M}_2}(\xi, \eta)\}$. Since $V_i^{\mathcal{M}_2} = V_i^{\mathcal{M}_1}$ (recall that $\lambda x, u V_i(x, u)$ is a formula that contains no element from S), $S_i^{\mathcal{M}_2}, S_i^{\mathcal{M}_3}$ coincide on $\{\eta \mid \neg V_i^{\mathcal{M}_1}(\xi, \eta)\}$. Thus $S_i^{\mathcal{M}_1}, S_i^{\mathcal{M}_3}$ coincide on $\{\eta \mid \neg V_i^{\mathcal{M}_1}(\xi, \eta)\}$.
3. $LS_{R(\xi)}(S_1^{\mathcal{M}_1}, S_1^{\mathcal{M}_2})$ and $LS_{R(\xi)}(S_1^{\mathcal{M}_2}, S_1^{\mathcal{M}_3})$, and thus (transitivity of $\subsetneq$) $LS_{R(\xi)}(S_1^{\mathcal{M}_1}, S_1^{\mathcal{M}_3})$.

Thus we proved all three requirements for $\mathcal{M}_1 \ll^{\xi} \mathcal{M}_3$ and transitivity is proved. Assymetry follows from transitivity and irreflexivity. ■

Recall that our notation is $Rx = \lambda u R(x,u)$ and that R is used to control the domain for the comparison of the minimized predicates. Intuitively, the following proposition says that if we always restrict the comparison of the minimized predicates to the same domain and there are finitely many models to the original theory, then the circumscription is satisfiable.

Proposition 4.4 *Let A be a satisfiable theory with finitely many models, and let $Rx = Ry$ for all $x, y \in |\mathcal{M}|^k$. Then, $C_{PS}[A; S_1/R; S_1/V_1, ..., S_n/V_n]$ is satisfiable.*

PROOF Assume there is no model $\mathcal{M}$ that satisfies $C_{PS}[A; S_1/R; S_1/V_1, ..., S_n/V_n]$. Then, there is no model $\mathcal{M}$ in $[A(S)]$ such that $\forall \mathcal{M}' \in [A(S)]\ \forall \xi \in |\mathcal{M}|^k\ \neg(\mathcal{M}' \ll^{\xi} \mathcal{M})$. Take $\xi_0 \in |\mathcal{M}|^k$. Since $[A]$ is finite and $\ll^{\xi_0}$ is a strict partial order, there must be at least one minimal element $\mathcal{M}$ for $\ll^{\xi_0}$ from the models of A. If a minimal model $\mathcal{M}_1$ of $\ll^{\xi_0}$ is not a minimal element according to some $\ll^{\xi_1}$, then $\ll^{\xi_1}$ has a minimal model $\mathcal{M}_2 \ll^{\xi_1} \mathcal{M}_1$. Since $\mathcal{M}_2$ is not minimal according to all of the $\ll$'s, there is yet another model $\mathcal{M}_3$ that is smaller according to yet another $\ll^{\xi_2}$.

Since there are finitely many models, if we continue such a chain of models, we will eventually end up with a cycle. Without loss of generality assume that the cycle is $\mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_n, \mathcal{M}_1$. But then we get $LS_{R(\xi_n)}(S_1^{\mathcal{M}_1}, S_1^{\mathcal{M}_n}), ...LS_{R(\xi_1)}(S_1^{\mathcal{M}_2}, S_1^{\mathcal{M}_1})$. Now we use the fact that $R(\xi_1) \equiv R(\xi_2) \equiv ... \equiv R(\xi_n)$ and get that $LS_{R(\xi_1)}(S_1^{\mathcal{M}_1}, S_1^{\mathcal{M}_n}), ...LS_{R(\xi_1)}(S_1^{\mathcal{M}_2}, S_1^{\mathcal{M}_1})$. Therefore, using the transitivity of $LS_{R(\xi_1)}$ we get $LS_{R(\xi_1)}(S_1^{\mathcal{M}_1}, S_1^{\mathcal{M}_1})$, contradicting the irreflexivity of $LS_{R(\xi_1)}$. ■

Last, we find out the following proposition.

Proposition 4.5 *1. McCarthy's Circumscription is equivalent to Point-Sensitive Circumscription with $R \equiv TRUE$ and $V \equiv TRUE$.*

2. Lifschitz's Generalized Pointwise Circumscription is equivalent to Point-Sensitive Circumscription with $Rx \equiv \lambda x'(x = x')$.

PROOF

1. $C_{PS}[A; S_1/TRUE; S_1/TRUE, ..., S_n/TRUE] = A(S) \wedge \forall x s \neg[LS_{TRUE}(s_1, S_1) \wedge \bigwedge_{i=1}^{n} EQ_{TRUE}(s_i, S_i) \wedge A(s)]$ s_1 is allowed to differ from S_1 with no limitations. Since $LS_{TRUE}(s_1, S_1) \equiv (s_1 \subsetneq S_1)$ we get the simplified Point-Sensitive Circumscription formula

$$A(S) \wedge \forall s \neg[s_1 < S_1 \wedge A(S)]$$

which is exactly McCarthy's Circumscription formula.

2. $C_{PS}[A; S_1/Rx; S_1/V_1, ..., S_n/V_n]$ is, by definition, $A(S) \wedge \forall x s \neg[LS_{Rx}(s_1, S_1) \wedge \bigwedge_{i=1}^{n} EQ_{V_i x}(s_i, S_i) \wedge A(s)]$. Let $Rx \equiv \lambda x'(x = x')$. Then, $LS_{Rx}(P, Q)$ is $\forall x'(Rx(x') \Rightarrow (Px' \Rightarrow Qx')) \wedge \exists x'(Rx(x') \wedge \neg Px' \wedge Qx')$ which is equivalent to $\neg Px \wedge Qx$. Thus, we get $C_{PS}[A; S_1/Rx; S_1/V_1, ..., S_n/V_n] \equiv C_{PW}[A; S_{i_0}; S_1/V_1, ..., S_n/V_n]$. ■

5 EXAMPLES USING POINT-SENSITIVE CIRCUMSCRIPTION

We examine two applications of Point-Sensitive Circumscription. First, we shortly re-examine the example from section 3. Then, we review one application of Pointwise Circumscription for the formalization of Prototypical Chronological Minimization (PCM), which is one known solution to the Frame Problem (for some ontological classes[3]). Finally, we examine this application with the proposed Point-Sensitive Circumscription and with Generalized Global Circumscription (as proposed in (Lifschitz, 1987) (formula (14))).

5.1 MULTIPLE MINIMAL MODELS

Let us first recall the example from section 3. Let $A(P) = (P(X) \vee P(Y)) \wedge X \neq Y$. Let $(V_P x) \equiv TRUE$ for all x, so that p is allowed to differ from P with no limitations. Let $R \equiv TRUE$, so that $LS_{Rx}(P, Q) \equiv (P \subsetneq Q)$. We get the simplified Point-Sensitive Circumscription formula

$$A(P) \wedge \forall p \neg[p < P \wedge A(p)]$$

which is exactly McCarthy's Circumscription formula. Thus, for $A(P) = (P(X) \vee P(Y)) \wedge X \neq Y$, we get the following result (contrary to proposition 3.1 for the Pointwise Circumscription case):

Proposition 5.1 *$C_{PS}[A(P); P/R; P/\lambda x.True]$ is equivalent to*

$$(\forall x(P(x) \Leftrightarrow x = X)) \vee (\forall x(P(x) \Leftrightarrow x = Y)) \wedge X \neq Y$$

PROOF First, notice that By theorem 4.5, $C_{PS}[A(P); P; P/\lambda x.True]$ is in fact equivalent to $A(P) \wedge \forall p \neg[p < P \wedge A(p)]$. At this point, all we need to notice is that we got McCarthy's Circumscription which is known to have the required equivalence. ■

Let us now re-examine the situation as seen by the semantics of proposition 4.2. Let $U = \{x, y\}$ be the set of el-

[3]See (Sandewall, 1994).

ements in the universe. We explore the same models discussed in section 3, M_X, M_Y, M_{XY} and $M_\emptyset$. Figure 3 below displays the two orders $\ll^X, \ll^Y$ on models.

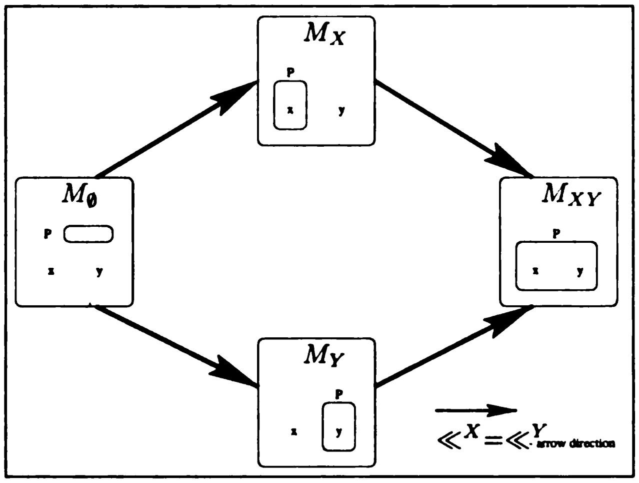


Figure 3: The orders $\ll^X$ and $\ll^Y$.

5.2 PCM USING POINTWISE CIRCUMSCRIPTION

Doherty and Lukaszewicz (Doherty and Lukaszewicz, 1994) examined various solutions to the Frame Problem, implementing them using either McCarthy's Circumscription or Lifschitz's Generalized Pointwise Circumscription (formula (2) above). We briefly review some of the theory of Features and Fluents (Sandewall, 1994) and one of the solutions that was examined in (Doherty and Lukaszewicz, 1994), namely, PCM, for which Pointwise Circumscription was used.

5.2.1 The general theory for PCM

The language $\mathcal{L}(FL)$ is a sorted first-order language with equality. There are two domain independent sorts, $\mathcal{T}$ for time points (t, s are variables of that sort) and $\mathcal{F}$ for propositional fluents (f, g are variables of that sort), which are functions from time to truth values. We include the predicate symbols $Holds, Clip$ of type $\mathcal{T} \times \mathcal{F}$ and the predicate symbols $<, \leq$ of type $\mathcal{T} \times \mathcal{T}$ (total orders on time points).

Let $\Gamma_C = \Gamma_{OBS} \cup \Gamma_{SCD} \cup \Gamma_{UNA}$ denote the set of formulas in $\mathcal{L}(FL)$ corresponding to the given scenario description, where Γ_{OBS} are the observation axioms (e.g., $Holds(f, s)$), Γ_{SCD} are the schedule axioms (a form of "effect axioms") and Γ_{UNA} are the appropriate unique name axioms for the fluent constants. Γ_{PER} denotes the set containing the two persistence axioms

$$\forall ft.Holds(t,f) \Rightarrow [Holds(t+1,f)\bar{\vee}Clip(t+1,f)]$$
$$\forall ft.\neg Holds(t,f) \Rightarrow [\neg Holds(t+1,f)\bar{\vee}Clip(t+1,f)]$$

PCM is defined roughly as follows (see (Sandewall and Shoham, 1994) for more details): Interpretations $I \ll_{pcm} I'$ iff $|I| = |I'|$ (the objects are the same) and there is t_0 such that

1. for all $t < t_0$, $R(t) = R'(t)$ (the fluents are identical in I and I' before t_0) and
2. $breakset(I, t_0) \subset breakset(I', t_0)$ (the fluent-value changes from $t_0 - 1$ to t_0 in I are a subset of the appropriate changes in I').

Then the set of models for PCM is $S_pcm(T) = Min(\ll_{pcm}, [T])$.

For the logic of PCM, we include the following. Let $\Gamma(Clip, Holds)$ denote the theory $\Gamma_C \cup \Gamma_{PER}$. Let $V(t,f) = \lambda t'f'.[\langle t,f\rangle = \langle t',f'\rangle \vee t < t']$ and $U(t,f) = \lambda t'f'.[t \leq t']$. Then, $C_{PW}[\Gamma(Clip, Holds); Clip; Clip/V, Holds/U]$ is

$$\begin{aligned}&\Gamma(Clip, Holds)\wedge\\ &\forall t,f.\forall clip, holds.\neg[Clip(t,f) \wedge \neg clip(t,f)\wedge\\ &\quad EQ_{V(t,f)}(Clip, clip) \wedge EQ_{U(t,f)}(Holds, holds)\wedge\\ &\quad \Gamma(clip, holds)]\end{aligned} \tag{5}$$

The intention is that when minimizing $Clip$ at time t for fluent f, the only part of $Clip$'s extension that will vary is the point $\langle t,f\rangle$ itself or $\langle t',f'\rangle$ for $t' > t$ and arbitrary f'. In addition, $Holds$ is varied only for time points $t' \geq t$.

5.2.2 YSP: global minimization and nondeterministic actions

Let us examine a variant of the Yale Shooting Scenario (YSS) ((Hanks and McDermott, 1986)). Assume that there are two turkeys, Turkey1 and Turkey2. As a result of the gun's being shot, exactly one turkey dies. For this, we have the propositions ALIVE1, DEAD1, ALIVE2, DEAD2, LOADED. Initially, both turkeys are alive, and the gun is LOADED. We have the action sequence of waiting and then shooting.

$$\begin{array}{ll} obs1 & [0,\infty)(ALIVE1 \leftrightarrow \neg DEAD1)\\ obs1 & [0,\infty)(ALIVE2 \leftrightarrow \neg DEAD2)\\ obs3 & [0]ALIVE1, ALIVE2, LOADED\\ scd1 & [\mathbf{s}]LOADED \rightarrow [\mathbf{s},\mathbf{t}]ALIVE1 := TF\\ obs4 & [\mathbf{s}]LOADED \rightarrow [\mathbf{s},\mathbf{t}]ALIVE2 \leftrightarrow \neg ALIVE1\\ scd2 & 0 < \mathbf{s} < \mathbf{t} \end{array} \tag{6}$$

$scd1$ says that, given that at time $\mathbf{s}$ the gun is loaded, then between $\mathbf{s}$ and $\mathbf{t}$, Turkey1 either dies or stays alive (as a result of the shooting action). $obs4$ then says that in the process of executing the turkeys (the time frame [s, t]), one turkey stays alive iff the other dies[4] (notice that we had to write *obs*4 as an observation because of the cues of the language). One of the intended results of applying (5) is that the turkeys are still alive at time s, but this is not what we get. Applying (5) yields two minimal models with

$$\begin{array}{lll} \mathcal{M}_1 \models & obs3 & [s]ALIVE1 \\ \mathcal{M}_2 \models & obs3 & [s]\neg ALIVE1 \end{array}$$

The reason for the unintended minimal model is the insistence on minimizing one fluent at a time, while not allowing other changes for other fluents for the same time point.

One proposed solution might be to minimize change for all the fluents at once for a given time point. For this solution, we set the policy V to be

$$V(t,f) = \lambda t' f'.[t \leq t']. \qquad (7)$$

Trying to use (7), we now encounter the problem discussed in section 3, i.e., that in the presence of fluents that are minimized in a circular fashion (i.e., minimizing P while varying Q and vice versa), the minimal models must agree on the involved predicates.

There are two intended minimal models for this scenario, one in which Turkey1 is eventually dead, and another in which Turkey2 is eventually dead. But $\leq^{\langle t, ALIVE1 \rangle}$ prefers Turkey2 to be dead, and $\leq^{\langle t, ALIVE1 \rangle}$ prefers Turkey2 to be dead. Thus, neither is a minimal model, and Pointwise Circumscription incurs inconsistency.

5.3 COMPARING SOLUTIONS

It is important to note that the problems we describe above do not affect the findings of (Doherty and Lukaszewicz, 1994), because these problems fall outside the ontological class that the formalization of (Doherty and Lukaszewicz, 1994) tried to formalize, namely the ontological class $\mathcal{K}p\text{-}IAex$. This class assumes that there are no observations outside the initial state. For the purposes of (Doherty and Lukaszewicz, 1994), a domain constraint such as $DEAD1 \leftrightarrow \neg ALIVE1$ is an observation for time points later than 0. Thus, to include this more *advanced* class, we need *filtering*[5].

Using Point-Sensitive Circumscription, we can do better than that. Let the range for comparison be controlled by $R(t,f) = \lambda t' f'[t' = t]$. As a result of applying the circumscription policy (7) to theory (6), we get

$$\begin{array}{l} C_{PS}[\Gamma(Clip, Holds); Clip/R; Clip/V, Holds/U] = \\ \quad \Gamma(Clip, Holds) \wedge \\ \quad \forall t, f \forall clip, holds \neg[LS_{R(t,f)}(clip, Clip) \wedge \\ \qquad EQ_{V(t,f)}(Clip, clip) \wedge EQ_{U(t,f)}(Holds, holds) \wedge \\ \qquad \Gamma(clip, holds)] \end{array} \qquad (8)$$

This formula minimizes $Clip$ for each time point for all fluents together (rather than per fluent). We get the following two minimal models

$$\begin{array}{lll} \mathcal{M}_1 \models & obs5 & [t1]\neg ALIVE1, DEAD1, \\ & & \neg ALIVE2, DEAD2, LOADED \\ \mathcal{M}_2 \models & obs5 & [t1]ALIVE1, \neg DEAD1, \\ & & \neg ALIVE2, DEAD2, LOADED \end{array}$$

which is exactly the outcome we wanted.

We do not claim that Point-Sensitive Circumscription can formalize the entire $\mathcal{K}\text{-}IA$ ontological class, but that it can take you farther than Pointwise Circumscription can. In more precise terms, we showed that Point-Sensitive Circumscription can simulate PCM in some cases that are outside $\mathcal{K}p - \text{-}IAex$ (the class treated by (Doherty and Lukaszewicz, 1994)). A hint that Point-Sensitive Circumscription may be able to simulate the ramification as given in the semantics of (Lin and Reiter, 1994), was given in (Amir, 1997).

Last, let us examine another proposal made in (Lifschitz, 1987), namely, an expanded version of *Global Circumscription*:

$$\begin{array}{l} Circ_G[A; P; Z/V] = \\ \quad A(P,Z) \wedge \forall pz \neg[EQ_V(z,Z) \wedge A(p,z) \wedge p < P] \end{array}$$

This is a reduced version of C_{PS} (taking $R \equiv TRUE$ $V(t,f) \equiv V$, we get $C_{PS} \equiv Circ_G$) in which we can not use different domains V_x for different points in the minimization. In the example above, it is essential to vary the fluents only for time points later than the current time point. Therefore, this proposal is not enough for the representation of the preference needed above.

6 CONCLUSIONS

In the context of Theories of Action, Pointwise Circumscription is sufficient if we restrict ourselves to deterministic actions and to cases in which there are no domain constraints and no action's effects are dependent on other actions' effects.

We showed that Pointwise Circumscription does not yield the right conclusions for some cases in which there are multiple minimal models (for McCarthy's Circumscription). This is due to the "point-wise-ness" of that form of circumscription. We described Point-Sensitive Circumscription,

[4]Meaning that exactly one turkey dies.

[5]Notice that our example (6) is in $\mathcal{K}\text{-}IAenx$.

a generalization of both McCarthy's Circumscription and Lifschitz's Pointwise Circumscription, and showed that it is useful for Theories of Action. Point-Sensitive Circumscription allows us to choose our degree of "point-wise-ness" (as suggested by the fact that both Circumscription and Pointwise Circumscription are its special cases) by controlling the domain used for the comparison of predicates. Since most uses of Pointwise Circumscription to date (e.g., (Doherty and Lukaszewicz, 1994)) refer mostly to the last property (variance control) rather than the first one ("point-wise-ness"), Point-Sensitive Circumscription is a useful substitute.

Acknowledgments

I wish to thank Leora Morgenstern and Tom Costello for encouraging me to expand my ideas in more detail. Tom Costello, Leora Morgenstern, Vladimir Lifschitz and John McCarthy commented on various versions of this paper. My deep thanks to all of them. Finally, I wish to thank Erika Henik for coining the term *Point-Sensitive*.

This research was supported by an ARPA (ONR) grant N00014-94-1-0775.

References

Amir, E. (1997). Formalizing Action using Set Theory and Pointwise Circumscription. In NRAC-97' and in http://www-formal.stanford.edu/eyal/nmr/foundation.ps.

Antoniou, G. (1997). *Nonmonotonic Reasoning*. MIT Press, Cambridge, Massachusetts.

Brewka, G. (1991). *Nonmonotonic Reasoning: Logical Foundations of Common Sense*. Cambridge University Press.

Brewka, G., Dix, J., and Konolige, K. (1997). *Nonmonotonic Reasoning: An Overview*, volume 73 of *CSLI Lecture Notes*. CSLI Publications.

D.M. Gabbay, C.J.Hogger, J., editor (1994). *Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 3: Nonmonotonic Reasoning and Uncertain Reasoning*. Oxford University Press, Great Britain.

Doherty, P. and Lukaszewicz, W. (1994). Circumscribing features and fluents. In Gabbay, D. and Ohlbach, H., editors, *Proceedings of the 1st int'l conf. on Temporal Logic*, pages 82–100.

Hanks, S. and McDermott, D. (1986). Default reasoning, nonmonotonic logics and frame problem. In *Proceedings of AAAI-86*, pages 328–333. Morgan Kaufmann.

Lifschitz, V. (1986). Pointwise circumscription: Preliminary report. In *Proc. AAAI-86*, pages 406–410.

Lifschitz, V. (1987). Pointwise circumscription. In Ginsberg, M., editor, *Readings in nonmonotonic reasoning*, pages 179–193. Morgan Kaufmann, San Mateo, CA.

Lifschitz, V. (1989). Circumscriptive theories: A logic-based framework for knowledge representation. In Thomason, R. H., editor, *Philosophical Logic and Artificial Intelligence*, pages 109–159. Kluwer Academic Publishers.

Lin, F. and Reiter, R. (1994). State constraints revisited. *Journal of Logic and Computation, Special Issue on Actions and Processes*.

McCarthy, J. (1980). Circumscription—A Form of Non-Monotonic Reasoning. *Artificial Intelligence*, 13:27–39. Reprinted in (McCarthy, 1990).

McCarthy, J. (1986). Applications of Circumscription to Formalizing Common Sense Knowledge. *Artificial Intelligence*, 28:89–116. Reprinted in (McCarthy, 1990).

McCarthy, J. (1990). *Formalization of common sense, papers by John McCarthy edited by V. Lifschitz*. Ablex.

McCarthy, J. (1998). Elaboration Tolerance. In Common-Sense '98 and in McCarthy's web page http://www-formal.stanford.edu/jmc/elaboration.html.

Pearl, J. (1990). Reasoning under uncertainty. *Annual Review of Computer Science*, 4:37–72.

Sandewall, E. (1994). *Features and Fluents*. Oxford University Press.

Sandewall, E. and Shoham, Y. (1994). Nonmonotonic Temporal Reasoning. In Gabbay, D., C.J.Hogger, and J.A.Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 4: Epistemic and Temporal Reasoning*. Oxford University Press.

Shanahan, M. (1997). *Solving the Frame Problem, a mathematical investigation of the common sense law of inertia*. MIT press, Cambridge, MA.

Planning I

Satisfiability Planning with Causal Theories

Norman McCain and **Hudson Turner**
Department of Computer Sciences
University of Texas at Austin
{mccain,hudson}@cs.utexas.edu

Abstract

A previous paper introduced the nonmonotonic formalism of *causal theories*, along with a general method for representing action domains in it. Here we show that causal action theories provide a basis for effective automated planning. To this end, we define several properties plans may have, such as executability, determinism, and validity. We then identify a class of causal action theories for which (i) there is a concise translation into classical logic, and (ii) the models of the resulting classical theories correspond to valid plans. These results enable satisfiability planning (in the sense of Kautz and Selman) on the basis of action formalizations that include indirect effects of actions (ramifications), implied action preconditions (qualifications), concurrent actions, and other features of action domains. Causal theories representing the large blocks world and logistics planning problems from [Kautz and Selman, 1996] are solved comparatively quickly, demonstrating the effectiveness of our approach.

1 INTRODUCTION

The language of causal theories is a simple nonmonotonic formalism, intended for representing the conditions under which facts are caused. The formalism was introduced in [McCain and Turner, 1997], along with a general method for representing action domains as causal theories. In the current paper, we describe an implemented approach to satisfiability planning [Kautz and Selman, 1992, 1996], which is based on a translation from the "definite" subclass of causal theories into classical propositional logic. This approach to planning is noteworthy for two reasons. First, it is based on a formalism for describing action domains that is more expressive than the STRIPS-based formalisms traditionally used in automated planning. Such features, for example, as indirect effects of actions (ramifications), implied action preconditions (qualifications), and concurrent actions are easily represented in causal theories.[1] Secondly, our experiments suggest that the additional expressiveness of causal theories comes with no performance penalty in satisfiability planning. Specifically, in this paper we show that the large blocks world and logistics planning problems used by Kautz and Selman [1996] to demonstrate the effectiveness of satisfiability planning can be conveniently represented as causal theories and solved in times comparable to those that they have obtained.

Because causal theories are more expressive than traditional planning languages, we must consider the preliminary question of when a sequence of actions is a valid plan for achieving a goal G in an initial situation S_0. A valid plan has two fundamental properties: sufficiency and executability. Roughly speaking, a sufficient plan will always achieve G if carried out starting in S_0, and an executable plan can always be carried out starting in S_0. We will make these ideas precise, in the setting of causal action theories.

We must also consider how to find valid plans by the satisfiability method. Assume that T is a classical propositional theory describing the worlds that are "causally possible" for an action domain. In satisfiability planning, a plan is obtained by extracting the sequence of actions from a model of T that satisfies both the initial state S_0 and the goal G. We will call a plan obtained in this way a causally possible plan, because what we know in this case is simply that there is at least one causally possible world in which the

[1]For more on the expressiveness of causal theories, see [McCain and Turner, 1997, Giunchiglia and Lifschitz, 1998, Lifschitz, 1997, McCain, 1997, Turner, 1998].

plan achieves G starting in S_0. In order for satisfiability planning to be sound, we must guarantee that the causally possible plans are in fact valid. Accordingly, we define a subclass of definite causal theories, called "simple," and show that their translations into classical logic are suitable for satisfiability planning. That is, the plans obtained from the models of their translations are not only causally possible, but also deterministic, and thus, as we will show, valid.

The main contributions of the paper are (1) to provide a theoretical foundation for satisfiability planning on the basis of causal theories, and (2) to present experimental evidence that the approach is relatively effective. More specifically, we define a family of fundamental properties a plan may have: causally possible, deterministic, sufficient, executable. We say a plan is valid if and only if it is sufficient and executable. We prove that every causally possible, deterministic plan is valid. We then identify a class of "simple" causal theories suitable for satisfiability planning. Simple causal theories have a concise translation into classical logic, and, as we prove, the classical models yield valid plans. Simple causal theories are very expressive, thus enabling planning with respect to a wide variety of action domains. We also provide experimental evidence that this planning approach can be very effective on classical problems, by solving, comparatively quickly, the large blocks worlds and logistics planning problems from [Kautz and Selman, 1996].

The paper is organized as follows. Section 2 reviews the syntax and semantics of causal theories, and defines a concise translation from definite causal theories into classical logic. Section 3 illustrates our method of formalizing action domains in causal theories. Section 4 defines plan validity and related notions for causal action theories. Section 5 defines the class of simple causal theories, and presents the main theorem showing that satisfiability planning is sound for simple theories. Section 6 describes an implementation of satisfiability planning with causal theories. Section 7 reports experimental results on the large blocks world and logistics planning problems from [Kautz and Selman, 1996]. Section 8 consists of the proof of the main theorem. We conclude briefly in Section 9.

2 CAUSAL THEORIES

2.1 SYNTAX

Begin with a language of propositional logic, whose signature is given by a nonempty set of atoms. (In application to formalizing action domains, atoms are taken to represent propositions about the values of fluents and the occurrences of actions at specific times.) Assume that the language includes a zero-place logical connective *True* such that *True* is a tautology. Let *False* stand for $\neg$*True*. A *literal* is an atom or the negation of an atom. For any literal L, $\overline{L}$ denotes its complement. We identify an interpretation with the set of literals true in it.

By a *causal law* we mean an expression of the form

$$\phi \Rightarrow \psi \qquad (1)$$

where ϕ and ψ are formulas of the underlying propositional language. By the *antecedent* and *consequent* of (1), we mean the formulas ϕ and ψ, respectively. We emphasize that (1) is not the material conditional $\phi \supset \psi$. The intended reading of (1) is: *Necessarily, if ϕ then the fact that ψ is caused.* Thus, we may say that (1) describes a condition under which ψ is caused.

By a *causal theory* we mean a set of causal laws.

2.2 SEMANTICS

For every causal theory D and interpretation I, let

$$D^I = \{\psi : \text{for some } \phi,\ \phi \Rightarrow \psi \in D \text{ and } I \models \phi\}.$$

Main definition. Let D be a causal theory. An interpretation I is *causally explained* according to D if I is the unique model of D^I.

As discussed in [McCain and Turner, 1997], this definition reflects a principle of "universal causation": every literal that is true in a causally explained interpretation I according to D is required to be caused in I according to D. Intuitively, when D describes an action domain, the interpretations that are causally explained according to D correspond to the world histories that are causally possible according to D. For a more helpful introduction to the causal theories formalism, see [McCain and Turner, 1997].

2.3 LITERAL COMPLETION

A causal theory D is *definite* if

- the consequent of each causal law in D is either a literal or *False*, and
- every literal is the consequent of finitely many causal laws in D.

Notice that, due to the first condition, an interpretation I is causally explained according to a definite causal theory D if and only if $I = D^I$.

In this paper, we will be particularly interested in definite causal theories, because they have a concise translation into classical propositional logic.

Let D be a definite causal theory. By the *literal completion* of D, denoted by $lcomp(D)$, we mean the classical propositional theory obtained by an elaboration of the Clark completion method [Clark, 1978], as follows. For each literal L in the language of D, include in $lcomp(D)$ the formula

$$L \equiv (\phi_1 \vee \cdots \vee \phi_n) \qquad (2)$$

where $\phi_1, \ldots, \phi_n$ are the antecedents of the causal laws in D with consequent L. (Of course, if no causal law in D has consequent L, then (2) becomes $L \equiv False$.) We will call formula (2) the *completion* of L. Also, for each causal law of the form $\phi \Rightarrow False$ in D, include in $lcomp(D)$ the formula $\neg\phi$. We will sometimes refer to causal laws with consequent *False* as *constraints*.

For example, let D_1 be the causal theory (in the language with exactly the atoms p and q) consisting of the causal laws

$$p \Rightarrow p\,,\ \neg q \Rightarrow p\,,\ q \Rightarrow q\,,\ \neg q \Rightarrow \neg q\,,\ q \Rightarrow False\,.$$

Causal theory D_1 is definite, and $lcomp(D_1)$ is

$$\{p \equiv p \vee \neg q\,,\ \neg p \equiv False\,,\ q \equiv q\,,\ \neg q \equiv \neg q\,,\ \neg q\}\,.$$

The following proposition generalizes slightly a result presented without proof in [McCain and Turner, 1997].

Proposition 1 *An intepretation I is causally explained according to a definite causal theory D if and only if I is a model of $lcomp(D)$.*

Proof. Assume $I = D^I$. Then for every literal $L \in I$, (i) there is a formula ϕ such that $\phi \Rightarrow L$ belongs to D and $I \models \phi$, and (ii) there is no formula ϕ such that $\phi \Rightarrow \overline{L}$ belongs to D and $I \models \phi$. It follows that for every literal $L \in I$, (i) I satisfies the completion of L, and (ii) I satisfies the completion of $\overline{L}$. That is, I satisfies the completion of every literal in the language of D. Similarly, since $False \notin D^I$, we can conclude that I satisfies every formula in $lcomp(D)$ obtained from a constraint. So I is a model of $lcomp(D)$. Proof in the other direction is similar. □

Let D be a causal theory, Γ a set of formulas, and ϕ a formula. We write

$$\Gamma \vdash_D \phi \qquad (3)$$

to say that ϕ is true in every model of Γ that is causally explained according to D.

The following corollary suggests an approach to query evaluation for definite causal theories.

Corollary 1 *Let D be a definite causal theory, Γ a set of formulas, and ϕ a formula. $\Gamma \vdash_D \phi$ if and only if $lcomp(D) \cup \Gamma \cup \{\neg\phi\}$ is unsatisfiable.*

3 CAUSAL ACTION THEORIES

3.1 $\mathcal{L}$(F,A,T) LANGUAGES

When representing an action domain by a causal theory, it is convenient to describe the underlying propositional signature by means of three pairwise-disjoint sets: a nonempty set $\mathbf{F}$ of *fluent names*, a set $\mathbf{A}$ of *action names*, and a nonempty set $\mathbf{T}$ of *time names* (corresponding to the natural numbers or an initial segment of the natural numbers). The atoms of the language $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ are divided into two classes, defined as follows. The *fluent atoms* are expressions of the form f_t such that $f \in \mathbf{F}$ and $t \in \mathbf{T}$. Intuitively, f_t is true if and only if the fluent f holds at time t. The *action atoms* are expressions of the form a_t such that $a \in \mathbf{A}$ and $t, t{+}1 \in \mathbf{T}$.[2] Intuitively, a_t is true if and only if the action a occurs at time t.

An *action literal* is an action atom or its negation. A *fluent literal* is a fluent atom or its negation. A *fluent formula* is a propositional combination of fluent atoms. We say that a formula refers to a time t if an atom of the form x_t occurs in it.

An $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ *domain description* is a causal theory in an $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ language.

3.2 EXAMPLE DOMAIN DESCRIPTIONS

To illustrate our approach to action formalization, we will represent a "falling dominos" domain and a "pendulum" domain. (For a gentler introduction, see [McCain and Turner, 1997].) These examples demonstrate interesting expressive possibilities of causal theories. Both descriptions are definite, and, as we will see, suitable for satisfiability planning.

3.2.1 Dominos Domain

We wish to describe the chain reaction of dominos falling over one after the other, after the first domino is tipped over.

Let the fluent names be $Up(1), \ldots, Up(4)$, and let the single action name be *Tip*. Identify time with the natural numbers $0, \ldots, 4$.

Here, as usual, we assume that facts about the occurrences of actions are *exogenous* to the causal theory. We express this assumption by writing the following schemas, where A is a meta-variable for action names. (Throughout, t is a meta-variable for time names.)

$$A_t \Rightarrow A_t \qquad (4)$$

$$\neg A_t \Rightarrow \neg A_t \qquad (5)$$

[2]As you might expect, the expression $t{+}1$ stands for the name of the successor of the number named by t.

Schema (4) says that the occurrence of an action A at a time t is caused whenever A occurs at t. Similarly, schema (5) says that the non-occurrence of A at time t is caused whenever A does not occur at t.

We also typically assume that facts about the initial values of fluents are exogenous, by writing the schemas

$$F_0 \Rightarrow F_0 \tag{6}$$
$$\neg F_0 \Rightarrow \neg F_0 \tag{7}$$

where F is a meta-variable for fluent names.

The fluent names $Up(1), \ldots, Up(4)$ will be designated *inertial.* For inertial fluents we write the following schemas, where I is a meta-variable for inertial fluent names.

$$I_t \wedge I_{t+1} \Rightarrow I_{t+1} \tag{8}$$
$$\neg I_t \wedge \neg I_{t+1} \Rightarrow \neg I_{t+1} \tag{9}$$

The first schema says that whenever an inertial fluent remains true from one time to the next, its truth at the latter time is caused. The second schema is similar. Taken together, these schemas solve the frame problem for inertial fluents.

We describe the direct effect and action precondition of the *Tip* action by writing

$$Tip_t \Rightarrow \neg Up(1)_{t+1} \tag{10}$$
$$\neg Up(1)_t \wedge Tip_t \Rightarrow False\,. \tag{11}$$

So *Tip* is the action of tipping over the first domino. It can only be done if the first domino is standing up.

We describe the chain reaction mechanism as follows, where d is a meta-variable ranging over numbers 1, 2, 3.

$$Up(d)_t \wedge \neg Up(d)_{t+1} \Rightarrow \neg Up(d+1)_{t+2} \tag{12}$$

Notice that this schema does not mention an action. It describes dynamic change involving three distinct time points. Roughly speaking, if domino d falls in the interval from t to $t+1$, then domino $d+1$ is caused to fall in the interval from $t+1$ to $t+2$.

Let D_2 be the causal theory given by schemas (4)–(12). Let I be the interpretation shown below.

Tip_0	$\neg Tip_1$	$\neg Tip_2$	$\neg Tip_3$	
$Up(1)_0$	$\bullet \neg Up(1)_1$	$\neg Up(1)_2$	$\neg Up(1)_3$	$\neg Up(1)_4$
$Up(2)_0$	$Up(2)_1$	$\bullet \neg Up(2)_2$	$\neg Up(2)_3$	$\neg Up(2)_4$
$Up(3)_0$	$Up(3)_1$	$Up(3)_2$	$\bullet \neg Up(3)_3$	$\neg Up(3)_4$
$Up(4)_0$	$Up(4)_1$	$Up(4)_2$	$Up(4)_3$	$\bullet \neg Up(4)_4$

Interpretation I specifies, for all actions a and times t such that $t+1 \in \mathbf{T}$, whether or not a occurs at t, and, for all fluents f and times t, whether or not f holds at t. The only action occurrence is *Tip* at time 0. One easily verifies that $I = D_2^I$. The four literals preceded by bullets appear in D_2^I due to the domain specific schemas (10) and (12). The other literals appear due to "standard schemas" (4)–(9). Hence, I is causally explained according to D_2.

3.2.2 Pendulum Domain

As a second example, we will formalize a dynamic domain from [Giunchiglia and Lifschitz, 1998]. In this domain there is a pendulum. In the course of nature (i.e., in the absence of interventions), the pendulum bob swings back and forth from right to left. However, at any time the agent can intervene to change the course of nature by holding the bob in its current location. So long as he continues to hold it, the bob remains where it is. When he no longer holds it, the bob resumes its natural course, swinging back and forth from right to left.

In formalizing the Pendulum domain, we will use a single action name *Hold* and fluent name *Right.* We will identify time with the natural numbers $0, \ldots, 4$.

The effects of the action *Hold* are specified straightforwardly by writing

$$Hold_t \wedge Right_t \Rightarrow Right_{t+1} \tag{13}$$
$$Hold_t \wedge \neg Right_t \Rightarrow \neg Right_{t+1}\,. \tag{14}$$

The behavior of the pendulum in the absence of interventions is described by writing

$$\neg Right_t \wedge Right_{t+1} \Rightarrow Right_{t+1} \tag{15}$$
$$Right_t \wedge \neg Right_{t+1} \Rightarrow \neg Right_{t+1}\,. \tag{16}$$

Like schemas (8) and (9) for inertia, schemas (15) and (16) describe a course of nature. Here the course of nature is dynamic rather than static, but otherwise there are clear similarities between the two pairs of schemas. Both pairs allow for the possibility that the course of nature may be overridden by the effects of actions, and both achieve this without mentioning facts about the non-occurrence of actions as preconditions.

In essence, schemas (15) and (16) solve the frame problem for the dynamic fluent *Right* in the same way that (8) and (9) solve the frame problem for inertial fluents.

Let D_3 be the causal theory given by schemas (4)–(7) and (13)–(16). Let the interpretation I be as follows.

$\neg Hold_0$	$Hold_1$	$Hold_2$	$\neg Hold_3$	
$Right_0$	$\neg Right_1$	$\neg Right_2$	$\neg Right_3$	$Right_4$

One easily verifies that $I = D_3^I$. Hence, I is causally explained according to D_3.

4 PLANNING WITH CAUSAL ACTION THEORIES

In this section we define fundamental notions related to planning, in the setting of causal action theories.

Let D be an $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description. By an ***initial state description*** we mean a set S_0 of fluent literals that refer to time 0 such that (1) for every fluent name $F \in \mathbf{F}$, exactly one of $F_0, \neg F_0$ belongs to S_0, and (2) $S_0 \not\vdash_D \mathit{False}$. Intuitively, an initial state description specifies an initial state that occurs in some causally possible world, i.e., a causally possible initial state. By a *time-specific goal*, we simply mean a fluent formula. Notice that a time-specific goal may refer to more than one time. By an ***action history*** we mean a set P of action literals such that, for every action name $A \in \mathbf{A}$ and time t such that $t+1 \in \mathbf{T}$, exactly one of $A_t, \neg A_t$ belongs to P. Every interpretation includes exactly one action history.

We will define when an action history P is a valid plan for achieving a time-specific goal G in an initial state S_0. This definition rests on the more fundamental notions of sufficiency and executability, which we also define. We define two other properties of plans, more naturally associated with satisfiability planning. One is determinism. The other is discussed next.

4.1 CAUSALLY POSSIBLE PLANS

Let D be an $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description, S_0 an initial state description, and G a time-specific goal. An action history P is a *causally possible* plan for achieving G in S_0 if

$$S_0 \cup P \not\vdash_D \neg G .$$

This condition says that there is, intuitively speaking, some causally possible world in which G can be achieved by executing P in initial state S_0.

Corollary 1 yields the following proposition.

Proposition 2 *Let D be a definite $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description, S_0 an initial state description, and G a time-specific goal. An action history P is included in a model of $lcomp(D) \cup S_0 \cup \{G\}$ if and only if P is a causally possible plan for achieving G in S_0.*

This proposition guarantees that every plan obtained by the satisfiability method is causally possible. Unfortunately, this is a rather weak guarantee. For example, in a nondeterministic "coin tossing" domain, a causally possible plan for having the coin lie heads at time 1, after lying tails at time 0, is simply to toss the coin at time 0. We can make this precise, as follows. We use a single fluent name *Heads*, a single action name *Toss*, and two times, 0 and 1. We designate *Heads* inertial. The causal theory D_4 for the action domain is represented by standard schemas (4)–(9), along with domain specific schemas (17) and (18) below.

$$\mathit{Toss}_t \wedge \mathit{Heads}_{t+1} \Rightarrow \mathit{Heads}_{t+1} \tag{17}$$

$$\mathit{Toss}_t \wedge \neg\mathit{Heads}_{t+1} \Rightarrow \neg\mathit{Heads}_{t+1} \tag{18}$$

Take $S_0 = \{\neg\mathit{Heads}_0\}$, $G = \mathit{Heads}_1$, and $P = \{\mathit{Toss}_0\}$. One easily checks that the interpretation $S_0 \cup P \cup \{G\}$ is causally explained according to D_4. Hence, P is a causally possible plan for achieving Heads_1 in S_0. On the other hand, P is also a causally possible plan for achieving $\neg\mathit{Heads}_1$ in S_0, since $S_0 \cup P \cup \{\neg G\}$ is also causally explained according to D_4.

4.2 SUFFICIENT PLANS

Let D be an $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description, S_0 an initial state description, and G a time-specific goal. An action history P is a ***sufficient*** plan for achieving G in S_0 if

$$S_0 \cup P \vdash_D G .$$

Intuitively, according to this definition, G will be achieved whenever P is done starting in S_0.

Sufficiency does not say anything about whether P can be executed in S_0, so it is not surprising that some sufficient plans are not valid. In fact, even plans that are both causally possible and sufficient can fail to be valid. Here is an example, again involving coin tossing, along with a second action of truly saying that the coin lies heads. We have a single fluent name, *Heads*, two action names, *Toss* and *TrulySayHeads*, and three times, 0, 1 and 2. Again *Heads* is inertial. The causal theory D_5 for this action domain is represented by standard schemas (4)–(9), together with domain specific schemas (17) and (18) from D_4, and one additional domain specific schema, shown below.

$$\mathit{TrulySayHeads}_t \wedge \neg\mathit{Heads}_t \Rightarrow \mathit{False} \tag{19}$$

Take $S_0 = \{\neg\mathit{Heads}_0\}$, $G = \mathit{Heads}_2$, and let P consist of $\mathit{Toss}_0, \neg\mathit{TrulySayHeads}_0, \neg\mathit{Toss}_1, \mathit{TrulySayHeads}_1$. So the plan is to toss the coin and then truly say heads. There is exactly one model of $S_0 \cup P$ that is causally explained by D_5—namely, the interpretation $S_0 \cup P \cup \{\mathit{Heads}_1, \mathit{Heads}_2\}$. Therefore, P is a sufficient, causally possible plan for achieving Heads_2 in S_0. That is, roughly speaking, there is a causally possible world in which doing P in S_0 achieves Heads_2, and, moreover, in any causally possible world in which P is done in S_0, Heads_2 is achieved. Nonetheless, P is not a valid plan. Intuitively, the problem is that P is not executable in S_0—it could be that the coin comes up tails after the initial toss, in which case the agent cannot truly say heads at time 1.

4.3 EXECUTABLE PLANS

We next define when a plan is executable in an initial state. Unfortunately, this condition is less convenient to state and check than the previous ones.

Let D be an $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description. For any time name $t \in \mathbf{T}$, let $\mathbf{T}|t = \{s \in \mathbf{T} : s \leq t\}$. Given a set X of $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ literals, and a time name t, we write $X|t$ to denote the set of all literals in X that belong to the restricted language $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T}|t)$.

Let P be an action history and S_0 an initial state description. We specify when $P|t$ is executable in S_0 by the following recursive definition. $P|0$ is *executable* in S_0. (Note that $P|0 = \emptyset$.) For all times $t+1 \in \mathbf{T}$, $P|t+1$ is *executable* in S_0 if the following two conditions hold: (i) $P|t$ is executable in S_0, and (ii) for every causally explained interpretation I that satisfies $S_0 \cup P|t$, there is a model of $I|t \cup P|t+1$ that is causally explained. Finally, we say that P itself is *executable* in S_0 if, for every time $t \in \mathbf{T}$, $P|t$ is executable in S_0.

So a plan P is executable if all of its prefixes are. Recall that a prefix $P|t$ completely specifies all action occurrences before time t, and that $P|t+1$ specifies in addition the action occurrences at time t. Thus $P|t+1$ is executable, roughly speaking, if $P|t$ is and, no matter the state of the world after executing $P|t$, the actions specified by P for time t can then be performed.

For example, consider more closely why the plan P from the last example is not executable in S_0. Recall that initially the coin lies tails. The prefix $P|1$ is executable in S_0. That is, it is possible to toss and not truly say heads at time 0. But prefix $P|2$ is not executable in S_0. Intuitively, it may not be possible to truly say heads at time 1. More precisely, notice that the interpretation I obtained from $S_0 \cup P|1$ by adding $\neg Heads_1, \neg Toss_1, \neg TrulySayHeads_1, \neg Heads_2$ is causally explained according to D_5, yet no model of $I|1 \cup P|2$ is causally explained. This is because no causally explained interpretation satisfies both $\neg Heads_1$ and $TrulySayHeads_1$.

4.4 VALID PLANS

Let D be an $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description, S_0 an initial state description, and G a time-specific goal. An action history P is a *valid* plan for achieving G in S_0 if it is both sufficient and executable.

The next proposition shows that valid plans are causally possible.

Lemma 1 *Let D be an $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description and S_0 an initial state description. If an action history P is executable in S_0, then there is model of $S_0 \cup P$ that is causally explained according to D.*

Proof Sketch. The definition of the executability of P in S_0 provides a basis for constructing a causally explained interpretation I such that, for all times $t \in \mathbf{T}$, I satisfies $S_0 \cup P|t$. □

Proposition 3 *Let D be an $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description, S_0 an initial state description, and G a time-specific goal. If P is a valid plan for achieving G in S_0, then it is a causally possible plan for achieving G in S_0.*

Proof. By Lemma 1, since P is executable in S_0, some model I of $S_0 \cup P$ is causally explained. Since P is sufficient for G in S_0, $S_0 \cup P \vdash_D G$. Hence, I satisfies G, which shows that $S_0 \cup P \not\vdash_D \neg G$. □

4.5 DETERMINISTIC PLANS

We will define one more class of plans, the deterministic plans. We will show that if a plan is causally possible and deterministic, it is valid. This is a key result in our approach to satisfiability planning. In Section 5 we will introduce the class of simple $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain descriptions, and show that for them all causally possible plans are deterministic, and thus valid.

Let D be an $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description, P an action history, and S_0 an initial state description. For every time t, $P|t$ is *deterministic* in S_0 if for all fluent names F and times $s \leq t$, $S_0 \cup P|t \vdash_D F_s$ or $S_0 \cup P|t \vdash_D \neg F_s$. We say that P is *deterministic* in S_0 if for every time $t \in \mathbf{T}$, $P|t$ is deterministic in S_0.

Thus a plan P is deterministic if all of its prefixes are. Recall that a prefix $P|t$ is a complete specification of action occurrences for all times before t. Prefix $P|t$ is deterministic if, roughly speaking, performance of the actions in $P|t$ starting in S_0 would completely determine the values of all fluents up to time t.

This definition yields a strong lemma.

Lemma 2 *Let D be an $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description and S_0 an initial state description. If an action history P is deterministic in S_0, then at most one model of $S_0 \cup P$ is causally explained according to D.*

Proof. Let I and I' be causally explained models of $S_0 \cup P$. Consider any fluent atom F_t. Since P is deterministic in S_0, so is $P|t$. Since both I and I' satisfy $S_0 \cup P|t$, it follows that they agree on F_t. Hence $I = I'$. □

The converse of Lemma 2 does not hold. P may fail to be deterministic in S_0 even when there is at most one causally explained model of $S_0 \cup P$. We illustrate this with another coin tossing example. Take a single fluent name, *Heads*, and three action names, *Toss*, *TrulySayHeads* and *TrulySayTails*. Identify time with the natural numbers. Once more we designate *Heads* inertial. The domain description D_6 is represented by standard schemas (4)–(9), together with domain specific schemas (17), (18) and (19) from D_5, and two more domain specific schemas, shown below.

$$TrulySayTails_t \wedge Heads_t \Rightarrow False \tag{20}$$
$$TrulySayHeads_t \equiv TrulySayTails_t \Rightarrow False \tag{21}$$

Due to (21), exactly one non-toss action occurs at every time in every causally possible world. Moreover, by (19) and (20), whenever truly say heads occurs, the coin lies heads, and whenever truly say tails occurs, the coin lies tails. Thus, each causally possible world is completely determined by its initial state and the actions that are performed in it. For instance, let $S_0 = \{\neg Heads_0\}$ and consider the plan P in which the agent initially tosses and concurrently truly says tails, and forever after truly says heads and does not toss. Although exactly one model of $S_0 \cup P$ is causally explained, P is not deterministic. This is because $P|1$ is not deterministic. That is, tossing and concurrently truly saying heads at time 0 simply does not determine the state of the coin at time 1.

Proposition 4 *Let D be an $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description, S_0 an initial state description, and G a time-specific goal. If P is a causally possible plan for achieving G in S_0 and P is also deterministic in S_0, then P is a valid plan for achieving G in S_0.*

Proof. Since P is a causally possible plan for achieving G in S_0, some model I^* of $S_0 \cup P \cup \{G\}$ is causally explained. By Lemma 2, no other model of $S_0 \cup P$ is causally explained. Since I^* satisfies G, P is sufficient for achieving G in S_0. To show that P is executable in S_0, we prove by induction that for all times t, $P|t$ is executable in S_0. The base case is trivial. For the inductive step, we show that $P|t+1$ is executable in S_0. By the inductive hypothesis, $P|t$ is executable in S_0. Thus we can complete the proof as follows. Assume that I is a causally explained model of $S_0 \cup P|t$. Notice that both I and I^* satisfy $S_0 \cup P|t$. Since P is deterministic in S_0, so is $P|t$, and it follows that $I^*|t = I|t$. Since I^* also satisfies $P|t+1$, we're done. □

Of course the converse of Proposition 4 does not hold, since valid plans need not be deterministic.

5 SATISFIABILITY PLANNING WITH CAUSAL THEORIES

In this section, we consider how to restrict definite $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain descriptions so that the causally possible plans are deterministic and thus, by Proposition 4, valid. To this end, we introduce the class of "simple" $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain descriptions.

5.1 SIMPLE DOMAIN DESCRIPTIONS

A definite $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description D is *simple* if it has the following three (yet to be defined) properties: it is inertially unambiguous, adequately acyclic, and respects the flow of time.

5.1.1 Inertially Unambiguous

Let $\mathbf{F}^+$ denote the set of all fluent atoms that refer to nonzero times. Causal laws in D of the form $\phi \wedge L \Rightarrow L$, where $L \in \mathbf{F}^+$ or $\overline{L} \in \mathbf{F}^+$, and ϕ is any $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ formula, will be called *inertia-like laws*.

Note that this definition covers not only causal laws obtained from standard inertia schemas (8)–(9) but also, for instance, causal laws such as those obtained from schemas (15)–(16) in the Pendulum domain D_3, which describe a dynamic course of nature. This definition also covers causal laws such as those obtained from schemas (17)–(18) in the coin-tossing domains. Although these causal laws express the direct nondeterministic effect of the coin-tossing action, they have a form similar to that of inertia laws.

We say that D is *inertially unambiguous* if it includes no pair of inertia-like laws

$$\phi \wedge F_{t+1} \Rightarrow F_{t+1} \tag{22}$$
$$\psi \wedge \neg F_{t+1} \Rightarrow \neg F_{t+1} \tag{23}$$

such that the formula $\phi \wedge \psi$ is satisfiable.

This exclusivity condition on ϕ and ψ is the only non-syntactic component of the definition of a simple domain description. Notice that the pairs of laws represented by schemas (8)–(9) for inertia and schemas (15)–(16) in the Pendulum domain satisfy this condition.

5.1.2 Adequately Acyclic

The *proper atom dependency graph* of D is the directed graph defined as follows. Its nodes are the atoms of the language of D. Let D' be the causal theory obtained from D by (i) deleting all causal laws whose consequent is *False*, and (ii) replacing each inertia-like law $\phi \wedge L \Rightarrow L$ with the causal law $\phi \Rightarrow L$. For each causal law in D', there is an edge from the atom that occurs in the consequent to each atom that occurs in the antecedent. We use the proper atom dependency graph to define an ordering on $\mathbf{F}^+$ as follows. For all $A, A' \in \mathbf{F}^+$, $A <_D A'$ if there is a nonempty path from A' to A. (So the edges in the graph point downward in the ordering.) We say that D is *adequately acyclic* if the ordering $<_D$ on $\mathbf{F}^+$ is well-founded.

Intuitively, this condition restricts cyclic causal dependencies between fluents, while allowing cycles that arise due to causal laws related to inertia.

5.1.3 Respects the Flow of Time

We say that D *respects the flow of time* if every causal law in D satisfies the following two conditions.

- If the consequent refers to a time t, then the antecedent does not refer to a time later than t.
- If the consequent is a fluent literal that refers to time t, then every action atom in the antecedent refers to a time earlier than t.

Notice that the description D_2 of the Dominos domain is simple, as is the description D_3 of the Pendulum domain. The coin-tossing domains D_4, D_5 and D_6 are not, because they are not inertially unambiguous.

5.2 SIMPLE DOMAIN DESCRIPTIONS YIELD VALID PLANS

Here is the main technical result related to simple domain descriptions. Its proof is postponed to Section 8.

Proposition 5 *Let D be a simple $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description, S_0 an initial state description, and G a time-specific goal. If P is a causally possible plan for achieving G in S_0, then P is a valid plan for achieving G in S_0.*

From this result, along with Propositions 2 and 3, we obtain the following characterization of satisfiability planning with simple $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain descriptions.

Theorem 1 *Let D be a simple $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description, S_0 an initial state description, and G a time-specific goal. An action history P is included in a model of*

$$lcomp(D) \cup S_0 \cup \{G\}$$

if and only if P is a valid plan for achieving G in S_0.

For effective satisfiability planning, we must of course also require that the simple $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description be finite, with $\mathbf{F}$, $\mathbf{A}$, and $\mathbf{T}$ finite as well.

6 SATISFIABILITY PLANNING PROGRAM

Given a finite signature and a set of schemas representing a finite, definite causal theory, it is straightforward to instantiate the schemas to obtain the represented (ground) causal theory, form its literal completion, and convert it to clausal form. We have written a Prolog program to carry out these tasks. It includes a procedure named `load_file/1`, which reads in a file such as the one displayed in Figure 1 for the Pendulum domain, and writes out in clausal form the literal completion of the causal theory. In the input syntax, fluent atoms f_t are represented as `h(`f`,`t`)`, and action atoms a_t are represented as `o(`a`,`t`)`. The symbols `h` and `o` are read as "holds" and "occurs," respectively.

```
% File: pendulum
:- declare_types  type(fluent,[right]),
   type(action,[hold]), type(time,[0..4]),
   type(atom,[h(fluent,time),o(action,time)]).
:- declare_variables  var(A,action),
   var(F,fluent), var([T,T1],time).
% domain specific schemas
   o(hold,T) & h(right,T) => h(right,T1)
                               where T1 is T+1.
   o(hold,T) & -h(right,T) => -h(right,T1)
                               where T1 is T+1.
   -h(right,T) & h(right,T1) => h(right,T1)
                               where T1 is T+1.
   h(right,T) & -h(right,T1) => -h(right,T1)
                               where T1 is T+1.
% standard schemas
   o(A,T) => o(A,T).   -o(A,T) => -o(A,T).
   h(F,0) => h(F,0).   -h(F,0) => -h(F,0).
```

Figure 1: Example Input File for the Planning System

After `load_file/1` has processed the domain description, planning problems are posed by calling the procedure `plan/0`, as shown in Figure 2. The procedure `plan/0` reads in an initial state description S_0 and a time-specific goal G, converts them to clausal form, and adds them to the clause set obtained from the domain description.[3] The resulting clause set is simplified, as in [Kautz and Selman, 1996][4], and submitted to the satisfiablity checker *rel_sat* [Bayardo and Schrag, 1997]. If $lcomp(D) \cup S_0 \cup \{G\}$ is satisfiable, *rel_sat* finds a satisfying interpretation and `plan/0` displays it, answering "yes." The plan P can be read off from this display. By Theorem 1, if D is simple, P is guaranteed to be a valid plan for achieving the goal G starting in initial state S_0. If *rel_sat* fails to find a sat-

[3]As illustrated in Figure 2, the initial state description S_0 can be replaced by a set Γ of formulas referring only to time 0 such that $\Gamma \vdash_D \phi$, where ϕ is the conjunction of the members of S_0, and yet $\Gamma \not\vdash_D False$.

[4]Steps: subsumption, unit propogation, subsumption.

isfying interpretation, plan/0 answers "no." Since the solver *rel_sat* is systematic, we know in this case that G cannot be achieved starting in S_0. In either case, the time spent in the solver *rel_sat* is reported.

```
| ?- load_file(pendulum).
% 9 atoms, 28 rules, 16 clauses loaded.
yes
| ?- plan.
enter facts and goal (then ctrl-d)
|: h(right,0).
|: -h(right,2) & h(right,4).
|:
0.  right
Actions: hold
1.  right
Actions:
2.  -right
Actions: hold
3.  -right
Actions:
4.  right
Elapsed Time (cpu sec): 0.01
yes
```

Figure 2: Planning Session with Pendulum Domain

7 LARGE PLANNING PROBLEMS

Here we report on the performance of our approach when applied to the large blocks world and logistics planning problems from [Kautz and Selman, 1996]. As far as we know, the results obtained there compare favorably with the best current general-purpose planning systems. We obtain comparable results.

7.1 BLOCKS WORLD PROBLEMS

The four blocks worlds planning problems from [Kautz and Selman, 1996] are characterized below.

```
Blocks World A.  9 blocks.  Requires 6 moves.
   Initial state: 2/1/0  4/3  8/7/6/5
   Goal state:    4/0  7/8/3  1/2/6/5
Blocks World B. 11 blocks.  Requires 9 moves.
   Initial state: 2/1/0  10/9/4/3  8/7/6/5
   Goal state:    0/4/9  7/8/3  1/2/10/6/5
Blocks World C. 15 blocks.  Requires 14 moves.
   Initial: 2/1/0/11/12  10/9/4/3/13/14  8/7/6/5
   Goal:    13/0/4/9  14/12/7/8/3  11/1/2/10/6/5
Blocks World D. 19 blocks.  Requires 18 moves.
   0/11/12 10/9/4/3/13/14 8/7/6/5 18/17/16/15/2/1
   16/17/18/13/0/4/9 14/12/7/8/3 11/1/2/15/10/6/5
```

Our input file representing Blocks World D is displayed in Figure 3. We adapt the "operator splitting" approach used by Kautz and Selman. Instead of axiomatizing an action $Move(b, l', l)$, they axiomatize three "component" actions, which we can write: $Pickup(b)$, $Takefrom(l')$, $Putat(l)$. Their axioms are based on Schubert's "explanation closure" [1990], augmented with state constraints. In comparison, we introduce names for only two components of the move action: $Pickup(b)$, $Putat(l)$. (When moved, a block is taken from where it currently is.) We also do not introduce a fluent $Clear(b)$. Kautz and Selman include in their description a number of state constraints that we omit.[5] Preliminary experiments indicated that additional state constraints in our blocks world descriptions increase solution times on larger problems.

One can easily verify that the causal theory represented in Figure 3 is simple. The main complication, compared to our descriptions of the Dominos and Pendulum domains involves action atoms, which are largely irrelevant in determining whether a description is simple. Here we include a family of action atoms that are "true by default" rather than exogenous. Thus, for example, the action *NoPickup* is assumed to occur, roughly speaking, and we describe the conditions under which it is caused not to occur—whenever $PickUp(B)$ occurs, for some block B. These auxiliary "inaction" atoms are used to stipulate that a pickup action occurs if and only if a putat action does.

7.2 LOGISTICS PLANNING PROBLEMS

The logistics domain is due to Veloso [1992]. Kautz and Selman studied three large logistics planning problems. Due to space constraints, we do not describe them, nor do we display an example input file.

The logistics domain is more complex than the blocks world domain. It includes several kinds of actions that can occur concurrently. Our description of the logistics domain does not use operator splitting (which is not generally applicable to concurrent actions). Preliminary experiments indicated that, in constrast to the blocks world, logistics domain descriptions should include a variety of state constraints in order to get consistently good performance. We note that the logistics domain description used in our experiments is simple, and thus suitable for satisfiablity planning.

7.3 EXPERIMENTAL RESULTS

In our experimental results on these planning problems, we report the size of the clausal theory obtained from the literal completion of the causal action

[5]Their state constraints still do not rule out all "physically impossible" states. This is in accordance with the usual practice in describing action domains for planning. Roughly speaking, one need only say enough to guarantee that no "illegal" state can be reached from a legal one. Intuitively, this is adequate because planning problems are posed in part by specifying a legal initial state.

```
:- declare_types  type(block,[0..18]), type(location,[block,table]),
     type(fluent,[on(block,location)]), type(action,[pickup(block),putat(location)]),
     type(inaction,[nopickup,noputat]), type(time,[0..18]),
     type(atom,[o(action,time),o(inaction,time),h(fluent,time)]).
:- declare_variables  var([B,B1],block), var([L,L1],location),
     var(F,fluent), var(A,action), var(X,inaction), var([T,T1],time).
% state constraints: the first two allow concise input of initial state and goal
     h(on(B,L),0) & h(on(B,L1),0) => false  where B \== L, B \== L1, L @< L1.
     h(on(B,L),18) & h(on(B,L1),18) => false  where B \== L, B \== L1, L @< L1.
     h(on(B,B),T) => false.
% direct effects of actions
     o(pickup(B),T) & o(putat(L),T) => h(on(B,L),T1)  where T1 is T+1, B \== L.
     h(on(B,L),T) & o(pickup(B),T) => -h(on(B,L),T1)  where T1 is T+1, B \== L.
% explicit action preconditions
     o(pickup(B),T) & h(on(B1,B),T) => false  where B =\= B1.
     o(putat(B),T) & h(on(B1,B),T) => false  where B =\= B1.
     o(pickup(B),T) & o(putat(B),T) => false.
     o(pickup(B),T) & o(putat(table),T) & h(on(B,table),T) => false.
% at most one move action at a time
     o(pickup(B),T) & o(pickup(B1),T) => false  where B @< B1.
     o(putat(L),T) & o(putat(L1),T) => false  where L @< L1.
     o(pickup(B),T) => -o(nopickup,T).   o(putat(L),T) => -o(noputat,T).
     o(nopickup,T) & -o(noputat,T) => false.   -o(nopickup,T) & o(noputat,T) => false.
% standard schemas
     h(F,0) => h(F,0).   -h(F,0) => -h(F,0).
     h(F,T) & h(F,T1) => h(F,T1)  where T1 is T+1.
     -h(F,T) & -h(F,T1) => -h(F,T1)  where T1 is T+1.
     o(A,T) => o(A,T).   -o(A,T) => -o(A,T).   o(X,T) => o(X,T).
```

Figure 3: Input File for Blocks World D

Table 1: Satisfiability Planning with Causal Theories
Sizes are for clausal theories obtained, via literal completion, from causal action theories (after simplification). Time in seconds in solver *rel_sat* on Sparcstation 5.

Instance	Atoms	Clauses	Literals	Time
BW A	383	2412	5984	0.13
BW B	934	6241	15903	0.81
BW C	2678	18868	48704	35.2
BW D	5745	41726	108267	620.0
LOG A	1643	9205	20712	3.7
LOG B	1760	10746	24134	8.4
LOG C	2300	14450	32346	25.0

Table 2: Kautz and Selman Problem Descriptions
Here we establish the benchmarks—the results for the clausal theories used in [Kautz and Selman, 1996], with solution times obtained in the same manner as in Table 1.

Instance	Atoms	Clauses	Literals	Time
BW A	459	4675	10809	0.20
BW B	1087	13772	31767	1.4
BW C	3016	50457	114314	66.3
BW D	6325	131973	294118	1052.0
LOG A	1782	20895	42497	2.5
LOG B	2069	29508	59896	9.8
LOG C	2809	48920	99090	32.3

theory—in terms of numbers of atoms, clauses and literal occurrences, after simplification—and time spent in the solver, following the reporting methodology of [Kautz and Selman, 1996]. Solution times are averaged over 20 runs of the solver *rel_sat* on a Sparcstation 5, using different random number seeds. Table 1 displays statistics for finding plans by our method.

For the sake of comparison, we performed the corresponding experiments on the problem descriptions from [Kautz and Selman, 1996], again using the solver *rel_sat* on a Sparcstation 5.[6] The results appear in Table 2. Bayardo and Schrag [1997] showed that, for the clausal theories of Kautz and Selman that we consider,

[6]Kautz and Selman considered two kinds of descriptions of the logistics domains (both in classical propositional logic): one based on intuitions underlying the planner Graphplan [Blum and Furst, 1995]; the other obtained by first describing the domain as in explanation closure, then eliminating all action atoms. In this second case, a satisfying interpretation does not include an action history. Rather it provides, as it were, a refinement of the planning problem. That is, the satisfying interpretation can be understood as an initial state and goal which together specify completely the values of all fluent atoms. In our reported results, we refer to the first kind of description. We note in comparison that the solver *rel_sat* takes longer for each instance of the second kind of description.

their solver *rel_sat* outperforms both of the solvers—one systematic, one stochastic—used in [Kautz and Selman, 1996].[7] Notice that in all cases except Logistics A our solution times are better.

Finally, in order to show that a plan is optimal (in the number of time steps), it is necessary to show that no shorter plan exists. For this purpose it is essential that a systematic solver be used. In Table 3, we report on the performance of our approach for this task, again using the solver *rel_sat*. For each problem, we report the time to fail to find a plan one step shorter than the optimal plan. Notice that, for these planning problems, the time needed to fail is comparable to the time needed to succeed.

Table 3: Proving Plans Optimal: Satisfiability Planning with Causal Theories. Here, in each case, the domain description includes one time step less than needed for a solution. Time reported is number of seconds required for solver *rel_sat* to determine unsatisfiability.

Instance	Atoms	Clauses	Literals	Time
BW A	281	1741	4211	0.04
BW B	788	5246	13276	0.43
BW C	2420	17033	43865	21.6
BW D	5343	38795	100544	374.2
LOG A	1354	7378	16595	2.2
LOG B	1498	8908	20026	31.3
LOG C	1946	11924	26710	54.8

8 PROOF OF PROPOSITION 5

We begin with the main lemma.

Lemma 3 *Let D be a definite $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description that is inertially unambiguous and adequately acyclic. Let P be an action history and S_0 an initial state description. At most one model of $S_0 \cup P$ is causally explained according to D.*

Proof. We proceed by the method of contradiction. Suppose that two distinct causally explained interpretations I and I' satisfy $S_0 \cup P$. Let X be the set of atoms on which I and I' disagree. Notice that X is a nonempty subset of $\mathbf{F}^+$, since I and I' differ, and yet agree on all atoms not in $\mathbf{F}^+$. Let X' consist of the members of X that are minimal (among members of X) with respect to the ordering $<_D$. Notice that X' is nonempty, since X is nonempty and $<_D$ restricted to X is well-founded. Finally, let F_{t+1} be a member of X' whose time subscript is minimal (among members of X'). Without loss of generality, assume that $I \models F_{t+1}$ and $I' \models \neg F_{t+1}$. Since $I = D^I$ and $I' = D^{I'}$, there must be a pair of causal laws $\phi \Rightarrow F_{t+1}$ and $\psi \Rightarrow \neg F_{t+1}$ in D such that $I \models \phi$ but $I' \not\models \phi$, and $I \not\models \psi$ but $I' \models \psi$. It follows that I and I' differ on at least one atom A that occurs in ϕ. Thus, $A \in X$ and also $A <_D F_{t+1}$. Consequently, by the minimality of F_{t+1}, A is F_{t+1}. Since D is adequately acyclic, $\phi \Rightarrow F_{t+1}$ must be of the form (22), and so can be written $\phi' \wedge F_{t+1} \Rightarrow F_{t+1}$. Since $I \models \phi$, $I \models \phi'$. A similar argument shows that $\psi \Rightarrow \neg F_{t+1}$ has form (23), and can be written $\psi' \wedge \neg F_{t+1} \Rightarrow \neg F_{t+1}$, with $I' \models \psi'$. Because D is inertially unambiguous, I' cannot satisfy both ϕ' and ψ'. Hence $I' \not\models \phi'$. (We complete the proof by showing that $I' \models \phi'$.) We have already shown that the only atom in ϕ on which I and I' differ is F_{t+1}, which is to say that the only atom in $\phi' \wedge F_{t+1}$ on which I and I' differ is F_{t+1}. Since D is adequately acylic, we know F_{t+1} does not occur in ϕ'. So I and I' agree on all atoms in ϕ', and since $I \models \phi'$, $I' \models \phi'$ as well. Contradiction. □

Let D be an $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description. For any $t \in \mathbf{T}$, let $D|t$ be the causal theory in the restricted language $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T}|t)$ consisting of all causal laws from D in that language.

Observe that if D is a simple $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description, then, for every time t, $D|t$ is a simple $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T}|t)$ domain description.

Lemma 4 *Let D be a simple $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description, S_0 an initial state description and P an action history. For all $t \in \mathbf{T}$, if I is a model of $S_0 \cup P|t$ that is causally explained according to D, then $I|t$ is the unique model of $S_0 \cup P|t$ causally explained according to $D|t$.*

Proof. Clearly $I|t$ is a model of $S_0 \cup P|t$. Given that D respects the flow of time, one easily verifies that $(D|t)^{I|t} = D^I|t$. Since $I = D^I$, $I|t = D^I|t$. So $I|t = (D|t)^{I|t}$, and we've shown that $I|t$ is a model of $S_0 \cup P|t$ that is causally explained according to $D|t$. We know that $D|t$ is simple since D is, so we can conclude by Lemma 3 that $I|t$ is unique. □

Lemma 5 *Let D be a simple $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description, S_0 an initial state description, and P an action history. If D has a causally explained interpretation satisfying $S_0 \cup P$, then P is deterministic in S_0.*

Proof. We need to show that, for all times $t \in \mathbf{T}$, $P|t$ is deterministic in S_0. Proof is by induction on t. The base case is trivial. By the inductive hypothesis, $P|t$ is deterministic in S_0. Assume that I and I' are models

[7]On the other hand, for their description of the logistics domain in which the action names are eliminated, their stochastic solver (properly tuned) is faster than *rel_sat*.

of $S_0 \cup P|t{+}1$ that are causally explained according to D. We need to show that $I|t{+}1 = I'|t{+}1$, which follows easily from Lemma 4. □

Proposition 4 and Lemma 5 yield Proposition 5.

9 CONCLUSION

This paper provides a theoretical foundation for satisfiability planning with causal theories. In our approach, action domain descriptions expressed as causal theories are translated into classical propositional logic. As we show, the classical models of the translation correspond exactly to the "causally possible" world histories according to the causal theory. Following Kautz and Selman, we then find plans by extracting them from models obtained by satisfiability checking.

In order to establish a basis upon which to judge the soundness of this approach to planning, we define a family of fundamental properties a plan may have: causally possible, deterministic, sufficient, executable. A plan is valid if and only if it is sufficient and executable. We observe that the plans obtained by the satisfiability method may, in general, fail to be sufficient or executable. They are only guaranteed to be causally possible. We show though that any causally possible plan that is deterministic is also valid.

We identify a class of "simple" domain descriptions for which the satisfiability method is applicable. Simple domain descriptions have a concise translation into classical logic. Moreover, we show that for such domains, the causally possible plans are deterministic and thus valid.

We describe an implemented satisfiability planning system based on these ideas, and provide experimental evidence that the approach can be computationally effective, by solving hard classical planning instances from [Kautz and Selman, 1996] comparatively quickly.

These developments are particularly noteworthy because of the expressive potential of simple causal theories, as illustrated by the Dominos and Pendulum domains presented in this paper. Thus, future applications of satisfiability planning with causal theories may address extensions to classical planning involving such features as concurrent actions and dynamic worlds.

Acknowledgments

We are grateful to our advisor, Vladimir Lifschitz. Thanks also to Enrico Giunchiglia. This work partially supported by NSF grant IRI-9306751. In addition, Norman McCain was partially supported by Texas Advanced Research Program grant 003658-242.

References

[Bayardo and Schrag, 1997] Roberto Bayardo and Robert Schrag. Using CSP look-back techniques to solve real-world SAT instances. In *Proc. of AAAI-97*, pages 203–208, 1997.

[Blum and Furst, 1995] A. Blum and M.L. Furst. Fast planning through planning graph analysis. In *Proc. IJCAI-95*, pages 1636–1642, 1995.

[Clark, 1978] Keith Clark. Negation as failure. In Herve Gallaire and Jack Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, 1978.

[Giunchiglia and Lifschitz, 1998] Enrico Giunchiglia and Vladimir Lifschitz. An action language based on causal logic. In Working Notes of 4th Symp. on Logical Formalizations of Commonsense Reasoning, 1998.

[Kautz and Selman, 1992] Henry Kautz and Bart Selman. Planning as satisfiability. In *Proc. of ECAI-92*, pages 359–379, 1992.

[Kautz and Selman, 1996] Henry Kautz and Bart Selman. Pushing the envelope: planning, propositional logic, and stochastic search. In *Proceedings of AAAI-96*, pages 1194–1201, 1996.

[Lifschitz, 1997] Vladimir Lifschitz. On the logic of causal explanation. *Artificial Intelligence*, 96:451–465, 1997.

[Lin, 1995] Fangzhen Lin. Embracing causality in specifying the indirect effects of actions. In *Proc. of IJCAI-95*, pages 1985–1991, 1995.

[McCain, 1997] Norman McCain. *Causality in Commonsense Reasoning about Actions.* PhD thesis, UT Austin, Dept. of Computer Sciences, 1997.

[McCain and Turner, 1995] Norman McCain and Hudson Turner. A causal theory of ramifications and qualifications. In *Proc. of IJCAI-95*, pages 1978–1984, 1995.

[McCain and Turner, 1997] Norman McCain and Hudson Turner. Causal theories of action and change. In *Proc. of AAAI-97*, pages 460–465, 1997.

[Schubert, 1990] Lenhart Schubert. Monotonic solution of the frame problem in the situation calculus: an efficient method for worlds with fully specified actions. In H.E. Kyburg, R. Loui, and G. Carlson, editors, *Knowledge Representation and Defeasible Reasoning*, pages 23–67. Kluwer, 1990.

[Turner, 1998] Hudson Turner. A logic of universal causation. To appear in *Artificial Intelligence*, 1998.

[Veloso, 1992] Manuela Veloso. *Learning by Analogical Reasoning in General Problem Solving.* PhD thesis, CMU, 1992. CS Technical Report CMU-CS-92-174.

On Measuring Plan Quality (A Preliminary Report)

Fangzhen Lin (flin@cs.ust.hk)
Department of Computer Science
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong

Abstract

By using an example from a robot navigating domain, we argue that to specify declaratively the behavior of an agent, we need to have a formal and explicit notion of "quality plans." To that end, we propose the following three domain independent measures of plan quality: a plan is said to be A-minimal if none of the actions in it can be deleted and have it continue to be a plan; it is said to be B-minimal if none of its segments can be deleted and have it continue to be a plan; and it is said to be C-minimal if none of its segments can be replaced by a single action and have it continue to be a plan. We show that given a plan, checking if it satisfies any of these three measures, and if not transforming it into one that does, can be done in polynomial numbers of plan tests. We also show how these three notions of plan minimality can be axiomatized naturally and straightforwardly in the situation calculus and, as a result, show their relationships with STRIPS triangle tables. Finally, we show their applications in the declarative specifications of agents' behaviours.

1 Introduction

Consider the classical SRI robot domain ([4]) where the robot can perform various chores such as delivering a bagel to someone in an office environment. Consider the requirement "close any doors that you open". There are many possible interpretations for this requirement. The easiest one is to take it as a "restore" goal of ([16]), which amounts to the constraint: if initially a door is closed, then by the end of the plan the door is still closed. But as Bacchus and Kabanza [2] note, this constraint is often too weak because it allows plans that leave some doors open longer than necessary.

To overcome this problem, Bacchus and Kabanza propose the following constraint: if the robot opens a door at any point of her plan, then at the next point in time, she must go through the door, and at the next point in time after that, she must close the door. While does the right thing for the version of the robot domain considered by Bacchus and Kabanza in [2][1], it does not work in general as it presupposes that the robot is able to close a door immediately after she has gone through it. For example, suppose that in order to open or close a door, the robot's hand must be empty. Then in order to go through an initially closed door while carrying an object, the robot will have to first find an appropriate place to deposit the object. Similarly, in order to close a door after going through it while carrying an object, she has to find a place to deposit the object as well. Thus, in a domain like this, an arbitrarily long time may elapse between the moment that the robot opens a door and the moment that she close it again.

This is where a formal notion of "high quality" plans can help: a more general solution along the line of Bacchus and Kabanza's is to require that after the robot has gone through an initially closed door, she must set out immediately to close it, that is, the next segment of the robot plan must be one that achieves "closed(door)" without any unnecessary actions, i.e. a "high quality" plan for "closed(door)".

There are many possible interpretations for what counts as a high quality plan. In this paper, we shall study three conceptually simple, computationally appealing, and domain independent ones. At the present time, we are not advancing any of them as the ultimate criteria of quality plans, if there is any such one, rather

[1] In this domain, the robot can always open and close a door as long as she is next to the door.

we are using them to illustrate the need for having an explicit and formal notion of quality plans.

This paper is organized as follows. In the next section, we lay out a general framework for studying plan quality. Then in section 3 we define three measures of plan quality and prove some results about their worst time complexity. In section 4 we axiomatize them in the situation calculus and relate them to regression ([10, 12]) and the STRIPS triangle tables ([3]). In section 5 we return to the robot domain and show how the aforementioned requirement about the statuses of doors can be axiomatized. In section 6, we discuss some related work, and finally in section 7, we make some concluding remarks.

2 Vocabularies and Notations

We assume the following vocabularies:

- A set $\mathcal{A}$ of actions.
- A set $\mathcal{G}$ of goals.
- The set $\mathcal{SA}$ of finite sequences of actions in $\mathcal{A}$. In particular, the empty sequence [] is in $\mathcal{SA}$.
- A binary predicate *Plan* on $\mathcal{SA} \times \mathcal{G}$. For any $\sigma \in \mathcal{SA}$ and any $g \in \mathcal{G}$, $Plan(\sigma, g)$, read "σ is a *plan* for g", means that the sequence of actions σ can be successfully executed in the (implicit) starting situation to achieve the goal g.

In the following, for any σ, γ and ξ in $\mathcal{SA}$, and any natural number n, we let $\sigma_\xi^{\gamma,n}$ to stand for the result of replacing the n-th subsequence γ in σ by ξ. If there are no such subsequence, then we let $\sigma_\xi^{\gamma,n}$ be σ. For example, using the Prolog list notation for finite sequences, we have

$$[a\ b\ c\ d]_{[]}^{[a],1} = [b\ c\ d],$$
$$[a\ b\ c\ d]_{[a\ a]}^{[a],2} = [a\ b\ c\ d],$$
$$[a\ b\ c\ d]_{[a]}^{[b\ c],1} = [a\ a\ d],$$
$$[a\ b\ c\ a\ b\ d]_{[b]}^{[a\ b],2} = [a\ b\ c\ b\ d]$$

where $a, b, c, d \in \mathcal{A}$, [] is the empty sequence.

3 Measuring plan quality

The usual measure on plan quality is based on the idea that the best plan is the shortest:[2]

> σ is an *optimal* plan for g if it is plan for g and there does not exist another plan γ for g such that the length of γ is smaller than that of σ.

In this paper we shall not concern ourself with optimal plans, for several reasons. Firstly, optimal plans are expensive to check and hard to generate. Secondly, optimality is a global notion. The information contained in a given suboptimal plan may have no help at all in "transforming it" into an optimal one: the problem is just the same as a fresh optimal plan generation problem. Computational issues aside, optimal plans are often not what we want for the purpose of specifying agents' behaviors, as we shall see in section 5.

Instead, in this paper we shall investigate the following three tractable and local measures of plan quality that are wearker than the notion of optimality:

1. A-minimality: a plan is A-minimal if none of the actions in it can be deleted and have it continue to be a plan.
2. B-minimality: a plan is B-minimal if none of its segments can be deleted and have it continue to be a plan.
3. C-minimality: a plan is C-minimal if none of its segments with lengths greater than 1 can be replaced by a single action and have it continue to be a plan.

Formally, we have:

Definition 1 *Let σ be a plan for g. We say σ is* A-minimal *if there do not exist an action $\alpha \in \mathcal{A}$ and a number n such that $\sigma_{[]}^{[\alpha],n}$ is not identical to σ but is a plan for g.*

We say σ is B-minimal *if there do not exist a sequence of actions $\gamma \in \mathcal{SA}$ and a number n such that $\sigma_{[]}^{\gamma,n}$ is not identical to σ but is a plan for g.*

We say σ is C-minimal *if there do not exist an action $\alpha \in \mathcal{A}$, a sequence of actions $\gamma \in \mathcal{SA}$ with length greater than 1, and a number n such that $\sigma_{[\alpha]}^{\gamma,n}$ is not identical to σ but is a plan for g.*

We want to point out that these three measures are by no means the only local and tractable (see below) ones. In particular, it is interesting to notice that C-minimality can be generalized in various ways. For instance, instead of $\sigma_{[\alpha]}^{\gamma,n}$, if we consider $\sigma_\xi^{\gamma,n}$ for all ξ whose length is less than that of γ, then C-minimality will be the same as optimality.

[2]Implicit in this definition is the assumption that each action have equal cost. This will also be the assumption that we will make in the rest of this paper. The extension to the more general case should be straightforward.

We now proceed to show some properties of these three minimality measures. The following proposition is immediate.

Proposition 1 *Let σ be a plan for g. If σ is B-minimal, then it is also A-minimal. If σ is C-minimal, then it is also B-minimal, provided the empty sequence is not a plan for g.*

It is not hard to see that there may be A-minimal but not B-minimal plans, and similarly there may be B-minimal but not C-minimal plans.

Let σ be a finite sequence of actions. We call testing if $Plan(\sigma, g)$ hold a *plan test* for σ with respect to g. For typical action specifications given by, for example ADL [11] or basic action theories (see below), if the starting situation is complete, then plan tests for a fixed goal can be done efficiently.

For all three versions of plan minimality, checking if a given plan is minimal, and if not, transforming it into a minimal one can be done in polynomial number of plan tests. Thus if plan tests are tractable, as we believe they often are, so will be the plan minimality tests.

Proposition 2 *Given a $\sigma \in \mathcal{SA}$ with length n and a goal g, testing if σ is an A-minimal plan for g can be done in $O(n)$ number of plan tests for action sequences whose lengths are no more than n.*

If σ is a plan for g but not an A-minimal one, then an A-minimal one can be constructed from σ in $O(n^2)$ number of plan tests for action sequences whose lengths are no more than n.

Proof: There are only n actions in σ, so testing if any of them can be deleted can be done in $O(n)$ number of plan tests. More precisely, consider the following Prolog-like rules:

```
aMinimal(X,[]) :- plan(X,g).
aMinimal(X, [a | Y]) :- append(X,Y,Z),
   not plan(Z,g), append(X,[a],X1),
   aMinimal(X1,Y).
```

where *append*$(X1, X2, X3)$ if the list $X3$ is the result of joining the lists $X1$ and $X2$. Clearly, σ is an A-minimal plan for g iff the query $aMinimal([], \sigma)$ returns "yes". The desired complexity result then follows from this program.[3]

If σ is not A-minmal, then we can repeatedly eliminate redundant actions one by one until none can be deleted anymore. Since there are only n actions, and finding one to delete takes $O(n)$ plan tests, so in the worst case, it takes $O(n^2)$ plan tests. Again, more precisely, consider the following Prolog-like rules:

```
aMinimal(X, [], X) :- plan(X,g).
aMinimal(X, [a | Y], Z) :- append(X,Y,Y1),
   plan(Y1,g), aMinimal([], Y1, Z).
aMinimal(X, [a | Y], Z) :- append(X,Y,Y1),
   not plan(Y1,g), append(X, [a], X1),
   aMinimal(X1, Y, Z).
```

Now given the query $aMinimal([], \sigma, X)$, the X returned, if any, will be an A-minimal plan for g. ■

Proposition 3 *Given a $\sigma \in \mathcal{SA}$ with length n and a goal g, testing if σ is a B-minimal plan for g can be done in $O(n^2)$ number of plan tests for action sequences whose lengths are no more than n.*

If σ is a plan for g but not a B-minimal one, then a B-minimal one can be constructed (from σ) in $O(n^3)$ number of plan tests for action sequences whose lengths are no more than n.

Proof: Similar to last proposition except now there are $O(n^2)$ segments to consider. ■

Proposition 4 *Let $\mathcal{A}$ be finite and of size m. Given a $\sigma \in \mathcal{SA}$ with length n and a goal g, testing if σ is a C-minimal plan for g can be done in $O(mn^2)$ number of plan tests for action sequences whose lengths are no more than n.*

If σ is a plan for g but not a C-minimal one, then a C-minimal one can be constructed (from σ) in $O(mn^3)$ number of plan tests for action sequences whose lengths are no more than n.

Proof: Given a plan σ of length n, there are $(n-2)^2$ segments with lengths greater than 1 to consider. For each of the segment, checking if it can be replaced by a single action takes $O(m)$ plan tests. So checking C-minimality takes $O(mn^2)$ plan tests.

If a plan is not C-minimal, then since finding a segment and an action to replace it takes $O(mn^2)$ plan tests, and we need to do such substitution at most n times because each substitution will decrease the length of the plan by at least 1, it takes at most $O(mn^3)$ plan tests to transform the plan into a C-minimal one. ■

These results are for the general case. For action theories that permit regressions, we can show some more

[3]Notice that *append* can be implemented in such a way (e.g. through pointers) that it takes only constant time.

concrete results. In this paper we shall consider action specifications that are given as *basic action theories* [8] in the situation calculus.

4 Plan quality in basic action theories

As far as the effects of actions are concerned, basic action theories have the same expressive power as ADL [11]. However, these action theories go well beyond being just a language for specifying the effects of actions. They are logical theories that can be used to reason about plans and planning processes. They can also be easily extended to include various domain constraints and control information [7, 6]. Before introducing basic action theories, we first briefly review the situation calculus, and define some necessary notations.

4.1 The situation calculus

Our version of the situation calculus [9] employs a many sorted second-order language. We assume the following sorts: ***situation*** for situations, ***action*** for actions, ***fluent*** for propositional fluents, and ***object*** for everything else.

We use the following domain independent predicates and functions:

- A distinct constant S_0 denoting the initial situation.
- A binary function *do* - for any action a and any situation s, $do(a,s)$ is the situation resulting from performing a in s.
- A binary predicate H - for any propositional fluent p and any situation s, $H(p,s)$ is true if p holds in s.
- A binary predicate $Poss$ - for any action a and any situation s, $Poss(a,s)$ is true if a is possible (executable) in s.
- A binary relation $\sqsubset$ on situations - for any situations s and s', $s \sqsubset s'$ if s' is the resulting situation of doing one or more actions in s. In the following, we consider $s \sqsubseteq s'$ to be a shorthand for $s \sqsubset s' \vee s = s'$.

We'll have a few other special predicate and functions, and will introduce them when they are needed.

We shall assume the following foundational axioms (cf. [14]):

$$S_0 \neq do(a,s), \tag{1}$$

$$do(a_1,s_1) = do(a_2,s_2) \supset (a_1 = a_2 \wedge s_1 = s_2), \tag{2}$$

$$(\forall P)[P(S_0) \wedge (\forall a,s)(P(s) \supset P(do(a,s))) \supset (\forall s)P(s)], \tag{3}$$

$$\neg s \sqsubset S_0, \tag{4}$$

$$s \sqsubset do(a,s') \equiv s \sqsubseteq s'. \tag{5}$$

The first two axioms are unique names assumptions about situations. They eliminate cycles, and avoid merging. The third axiom is second order induction. It amounts to the domain closure axiom that every situation has to be obtained from the initial one by repeatedly applying the function *do*. The last two axioms define $\sqsubset$ inductively.

By our foundational axioms, for any situation s, there must be a unique finite sequence of actions σ such that $s = do(\sigma, S_0)$, where for any s', any action a, and any sequence L, we define:[4]

$$do([], s') \overset{def}{=} s', \quad do([a|L], s') \overset{def}{=} do(L, do(a,s')).$$

This means that by our foundational axioms, the set of situations is isomorphic to the set of finite sequences of action. This isomorphism will be the logical basis on which properties of plans are axiomatized in the situation calculus.

4.2 Fluent formulas

We will find it convenient to talk about *fluent formulas*, which are generalizations of those in [15]. Informally, a fluent formula is a situation calculus formula with situation terms suppressed. Formally, we define them as follows:

1. Any formula that does not mention a situation term is a fluent formula.
2. If F is a n-ary fluent, and $t_1, ..., t_n$ are object terms, then $F(t_1, ..., t_n)$ is a fluent formula.
3. If ϕ and φ are fluent formulas, x an object variable, and p a fluent variable, then $\neg\varphi$, $\varphi \vee \phi$, $(\forall x).\varphi$ and $(\forall p).\phi$ are fluent formulas. (Other connectives such as $\wedge$, $\supset$, $\equiv$, and $\exists$ can be defined as usual.)

For example, $A1 \neq A2$, $on(block1, block2)$, $(\forall p)(p \supset p)$ are fluent formulas, where $A1$ and A are action constants and *on* a binary fluent.

[4]We write $e1 \overset{def}{=} e2$ if $e1$ is considered to be a shorthand for $e2$.

Given a fluent formula φ and a situation term S, we define $H(\varphi, S)$ to be a shorthand for a situation calculus formula according to the following rules:

$$\begin{aligned}
&H(\varphi, S) \stackrel{def}{=} \varphi, \text{ provided } \varphi \text{ does not mention any situation terms}\\
&H(\neg\varphi, S) \stackrel{def}{=} \neg H(\varphi, S),\\
&H(\varphi \vee \phi, S) \stackrel{def}{=} H(\varphi, S) \vee H(\phi, S),\\
&H((\forall x)\varphi, S) \stackrel{def}{=} (\forall x)H(\varphi, S),\\
&H((\forall p)\varphi, S) \stackrel{def}{=} (\forall p)H(\varphi, S).
\end{aligned}$$

For example,

$$\begin{aligned}
&H(A_1 \neq A_2, s) \stackrel{def}{=} A_1 \neq A_2,\\
&H(on(B1, B2) \vee clear(B2), s) \stackrel{def}{=}\\
&\quad H(on(B1, B2), s) \vee H(clear(B2), s),\\
&H((\forall x)[clear(x) \equiv \neg(\exists y)on(y, x)], s) \stackrel{def}{=}\\
&\quad (\forall x)[H(clear(x), s) \equiv \neg(\exists y)H(on(y, x), s)].
\end{aligned}$$

4.3 Plans and minimal plans

We now proceed to axiomatize the three versions of plan minimality in the situation calculus.

First of all, we take a *goal* to be a ground fluent formula. Secondly, instead of finite sequences of actions, we consider situations. Because of the isomorphism between situations and finite sequences of actions that we mentioned above, we don't lose anything by doing this. This change of ontology is important here because situations are first class objects in the situation calculus, while finite sequences of actions are not. Finally, we make explicit the starting situation in which a plan is to be executed. Therefore, instead of a binary *Plan* predicate, we extend it to a ternary one: for any goal g, situations s_1 and s_2, $Plan(g, s_1, s_2)$ if the sequence of actions that takes s_1 (the starting situation) to s_2 (the goal situation) is executable in s_1 and makes g true afterward:

$$\begin{aligned}
&Plan(g, s_1, s_2) \stackrel{def}{=} H(g, s_2) \wedge s_1 \sqsubseteq s_2 \wedge\\
&\quad (\forall a, s)[s_1 \sqsubset do(a, s) \sqsubseteq s_2 \supset Poss(a, s)].
\end{aligned}$$

Corresponding to the three versions of plan minimality defined earlier, we have the following three macros:

$$\begin{aligned}
&Aminimal(g, s_1, s_2) \stackrel{def}{=} Plan(g, s_1, s_2) \wedge\\
&\quad (\forall a, s)[s_1 \sqsubset do(a, s) \sqsubseteq s_2 \supset\\
&\qquad \neg Plan(g, s_1, del(s_2, s, do(a, s)))],\\
&Bminimal(g, s_1, s_2) \stackrel{def}{=} Plan(g, s_1, s_2) \wedge\\
&\quad (\forall s', s'')[s_1 \sqsubseteq s' \sqsubset s'' \sqsubseteq s_2 \supset\\
&\qquad \neg Plan(g, s_1, del(s_2, s', s''))],\\
&Cminimal(g, s_1, s_2) \stackrel{def}{=} Plan(g, s_1, s_2) \wedge\\
&\quad (\forall a, s', s'', s^*)[s_1 \sqsubseteq s' \sqsubset s^* \sqsubset s'' \sqsubseteq s_2 \supset\\
&\qquad \neg Plan(g, s_1, repl(s_2, s', s'', a))],
\end{aligned}$$

where for any situations s, s_1, s_2, $del(s, s_1, s_2)$, the situation resulted from deleting the segment of s between s_1 and s_2, is the unique situation that satisfies the following axioms:

$$\neg(s_1 \sqsubseteq s_2 \sqsubseteq s) \supset del(s, s_1, s_2) = s, \tag{6}$$

$$s_1 \sqsubseteq s_2 \supset del(s_2, s_1, s_2) = s_1, \tag{7}$$

$$\begin{aligned}&s_1 \sqsubseteq s_2 \sqsubset do(a, s) \supset\\ &\quad del(do(a, s), s_1, s_2) = do(a, del(s, s_1, s_2)),\end{aligned} \tag{8}$$

and for any situations s, s_1, s_2, and any action a, $repl(s, s_1, s_2, a)$, the result of replacing the segment between s_1 and s_2 in s by the action a, is the unique situation that satisfies the following axioms:

$$\neg(s_1 \sqsubset s_2 \sqsubseteq s) \supset repl(s, s_1, s_2, a) = s, \tag{9}$$

$$s_1 \sqsubset s_2 \supset repl(s_2, s_1, s_2, a) = do(a, s_1), \tag{10}$$

$$\begin{aligned}&s_1 \sqsubset s_2 \sqsubset do(b, s) \supset\\ &repl(do(b, s), s_1, s_2, a) = do(b, repl(s, s_1, s_2, a)).\end{aligned} \tag{11}$$

In the following, we shall denote by $\mathcal{D}_{min}$ the set of axioms (6) – (11).

4.4 Basic action theories

A *basic action theory* $\mathcal{D}$ is one that has the following form: (cf. Reiter [13] and Lin and Reiter [8]):

$$\mathcal{D} = \Sigma \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0},$$

where

- Σ is the set of the foundational axioms (1) – (5).
- $\mathcal{D}_{ss}$ is a set of successor state axioms, one for each fluent F, of the form:

  $$Poss(a, s) \supset H(F(\vec{x}), do(a, s)) \equiv H(\Phi_F(\vec{x}, a), s), \tag{12}$$

  where $\Phi_F(\vec{x}, a)$ is a fluent formula whose free variables are among $\vec{x}, a$. Intuitively, a successor state axiom for F defines the truth value of F in a successor situation in terms of the truth values of fluents in the current situation.
- $\mathcal{D}_{ap}$ is a set of action precondition axioms of the form:

  $$Poss(A(\vec{x}), s) \equiv H(\Psi_A(\vec{x}), s), \tag{13}$$

where A is an action, and $\Psi_A(\vec{x})$ is a fluent formula whose free variables are among $\vec{x}$.

- $\mathcal{D}_{una}$ is the set of unique names axioms for actions.
- $\mathcal{D}_{S_0}$, the initial database, is a finite set of first-order sentences of the form $H(\phi, S_0)$, where ϕ is a ground fluent formula.

Basic action theories are computationally appealing because they permit reasoning by regression:

Definition 2 (Regression) *Let $\mathcal{D}$ be a basic action theory. Let g be a fluent formula, and α an action term of the form $A(\vec{t})$. The regression of g over α (under $\mathcal{D}$), written $R(g, \alpha)$, is the fluent formula obtained from g by replacing every expression of the form $F(\vec{u})$ in it by $\Phi_F(\vec{u}, A(\vec{t}))$, where Φ_F is as in the successor state axiom (12) for F.*

The following is the fundamental property of regression:

Proposition 5 *Let $\mathcal{D}$ be a basic action theory, then*

$$\mathcal{D} \models (\forall s).Poss(\alpha, s) \supset H(g, do(\alpha, s)) \equiv H(R(g, \alpha), s).$$

The proof of a proposition similar to this one can be found in (Reiter [14]).

4.5 Minimal plans under basic action theories

We now show how to check if a given plan is a minimal one using regression under a basic action theory.

Given a basic action theory $\mathcal{D}$, a goal g, and a finite sequence of actions $[\alpha_1, ..., \alpha_n]$, let

$$g_{n+1} = g,$$
$$g_k = R(g_{k+1}, \alpha_k) \wedge \Psi_A(\vec{t}), \text{ for } 1 \le k \le n$$

where $\alpha_k = A(\vec{t})$ for some action A and tuple $\vec{t}$, and Ψ_A is as in (13) in $\mathcal{D}$.

The intuition behind g_k is that it is the condition under which the sequence of actions $[\alpha_k, ..., \alpha_n]$ can be successfully executed to achieve the goal g:

Lemma 1 *Under the basic action theory $\mathcal{D}$, we have:*

$$(\forall s).H(g_k, s) \equiv Plan(g, s, do([\alpha_k, ..., \alpha_n], s)).$$

Proof: By the definition of *Plan*, we need to prove the following equivalence under $\mathcal{D}$:

$$(\forall s).H(g_k, s) \equiv$$
$$H(g, do([\alpha_k, ..., \alpha_n], s)) \wedge Poss([\alpha_k, ..., \alpha_n], s),$$

where $Poss([\alpha_k, ..., \alpha_n], s)$ is defined as follows:

$$Poss([], s) \overset{def}{=} true,$$
$$Poss([\beta, \beta_1, ..., \beta_m], s) \overset{def}{=}$$
$$Poss([\beta_1, ..., \beta_m], do(\beta, s)) \wedge Poss(\beta, s).$$

We prove this by induction on k, with $k = n + 1$ as the base case, which is trivial:

$$(\forall s).H(g_{n+1}, s) \equiv H(g, do([], s)) \wedge Poss([], s).$$

Inductively, assume that we have

$$(\forall s).H(g_k, s) \equiv$$
$$H(g, do([\alpha_k, ..., \alpha_n], s)) \wedge Poss([\alpha_k, ..., \alpha_n], s),$$

for $1 \le k = m \le n + 1$. We show that this is the case for $k = m - 1$ as well. Assuming that α_{m-1} is $A(\vec{t})$, we have

$$H(g_{m-1}, s) \equiv$$
$$H(\Psi_A(\vec{t}), s) \wedge H(R(g_m, \alpha_{m-1}), s) \equiv$$
$$Poss(\alpha_{m-1}, s) \wedge H(R(g_m, \alpha_{m-1}), s) \equiv$$

by the basic property of regression

$$Poss(\alpha_{m-1}, s) \wedge H(g_m, do(\alpha_{m-1}, s)) \equiv$$

by the inductive assumption

$$Poss(\alpha_{m-1}, s) \wedge Poss([\alpha_m, ..., \alpha_n], do(\alpha_{m-1}, s)) \wedge$$
$$H(g, do([\alpha_m, ..., \alpha_n], do(\alpha_{m-1}, s))) \equiv$$
$$Poss([\alpha_{m-1}, ..., \alpha_n], s) \wedge H(g, do([\alpha_{m-1}, ..., \alpha_n], s)).$$

This proves the inductive step, thus the lemma. ■

The sequence of fluent formulas $g_1, ..., g_{n+1}$ captures the essence of the triangle table [3] of a plan: g_i is essentially the i-th kernel of the plan. Just as STRIPS uses kernels to control plan execution, we can use these fluent formulas to check various notions of plan minimality. We show here how this can be done for B-minimality. Similar results hold for A-minimality and C-minimality.

Theorem 1 *Let $\mathcal{D}$ be a basic action, g a goal, and $[\alpha_1, ..., \alpha_n]$ a finite sequence of ground action terms. Under $\mathcal{D} \cup \mathcal{D}_{min}$, we have:*

$$Plan(g, s, do([\alpha_1, ..., \alpha_n], s)) \supset$$
$$Bminimal(g, s, do([\alpha_1, ..., \alpha_n], s)) \equiv$$
$$\{ \bigwedge_{0 \le k < n,\ k+2 \le j \le n+1} \neg H(g_j, do([\alpha_1, ..., \alpha_k], s))\}.$$

Proof: It is more convenient to prove the following equivalent sentence:

$$Plan(g, s, do([\alpha_1, ..., \alpha_n], s)) \supset$$
$$\neg Bminimal(g, s, do([\alpha_1, ..., \alpha_n], s)) \equiv$$
$$\{ \bigvee_{0 \le k < n,\ k+2 \le j \le n+1} H(g_j, do([\alpha_1, ..., \alpha_k], s))\}.$$

Suppose $Plan(g, s, do([\alpha_1, ..., \alpha_n], s))$.

$\Rightarrow$:

Assume

$$\neg Bminimal(g, s, do([\alpha_1, ..., \alpha_n], s)).$$

Then by the definition of *Bminimal*, there must be two situations s' and s'' such that

$$s \sqsubseteq s' \sqsubset s'' \sqsubseteq do([\alpha_1, ..., \alpha_n], s)$$

and

$$Plan(g, s, del(do([\alpha_1, ..., \alpha_n], s), s', s'')).$$

By our foundational axioms in Σ, there must be $0 \le i < j \le n$ such that

$$s' = do([\alpha_1, ..., \alpha_i], s),\ \ s'' = do([\alpha_1, ..., \alpha_j], s)$$

By $\mathcal{D}_{min}$, this means that

$$del(do([\alpha_1, ..., \alpha_n], s), s', s'')) =$$
$$do([\alpha_1, ..., \alpha_i, \alpha_{j+1}, ..., \alpha_n], s).$$

Therefore

$$Plan(g, s, do([\alpha_1, ..., \alpha_i, \alpha_{j+1}, ..., \alpha_n], s))$$

Thus

$$Plan(g, s_i, do([\alpha_{j+1}, ..., \alpha_n], s_i))$$

where

$$s_i = do([\alpha_1, ..., \alpha_i], s).$$

So by Lemma 1, we have $H(g_{j+1}, s_i)$. This completes the proof of "$\Rightarrow$".

$\Leftarrow$:

Assume that for some $0 \le k < n$, and $k+2 \le j \le n+1$, we have

$$H(g_j, s_k),$$

where $s_k = do([\alpha_1, ..., \alpha_k], s)$. By Lemma 1, we have

$$Plan(g, s_k, do([\alpha_j, ..., \alpha_n], s_k)).$$

Together with $Plan(g, s, do([\alpha_1, ..., \alpha_n], s))$, we have

$$Plan(g, s, do([\alpha_1, ..., \alpha_k, \alpha_j, ..., \alpha_n], s).$$

Since

$$del(do([\alpha_1, ..., \alpha_n], s), s_k, s'')) =$$
$$do([\alpha_1, ..., \alpha_k, \alpha_j, ..., \alpha_n], s),$$

where

$$s'' = do([\alpha_1, ..., \alpha_{j-1}], s),$$

we conclude that $\neg Bminimal(g, s, do([\alpha_1, ..., \alpha_n], s))$. This completes the proof of "$\Leftarrow$". ■

5 The door please, Shakey

To show the usefulness of a logical formalization of plan quality, we now return to the robot domain that we discussed earlier in the paper. Our aim is to make precise the requirement "close those doors that you open." As we mentioned, there are several ways to interpret this requirement. We start with the easiest, taking it as a "restore" goal of [16]:

$$(\forall d).H(closed(d), S_0) \supset H(closed(d), S_{end}), \quad (14)$$

where $S_{end} = do(\sigma, S_0)$, σ the given plan, and *closed*(d) a fluent meaning that door d is closed.

Axiom (14) says that if a door is initially closed, then after the plan is done, it must still be closed.[5] As we discussed earlier, this constraint is often too weak as it allows plans that leave a door open longer than necessary, even if we insist on σ being, for example, an optimal plan.

There are several ways to strengthen this constraint. One is to require the robot to immediately set out to close an initially closed door right after *the last time* she went through it:

$$(\forall d, s).\{s \sqsubseteq S_{end} \wedge H(through(d), s) \wedge$$
$$(\forall s')[s \sqsubset s' \sqsubseteq S_{end} \supset \neg H(through(d), s')]\} \supset$$
$$(\exists s_1)(s \sqsubseteq s_1 \sqsubseteq S_{end} \wedge \quad (15)$$
$$Bminimal(closed(d), s, s_1)),$$

here *through*(d) is a fluent which is true right after the robot has gone through door d and false at all other times. Notice that we use *Bminimal* in the above axiom to stand for "quality plans". Alternatively, one can use *Aminimal*, *Cminimal*, or any other measure one might have. In general, the stronger the measure, the tighter the resulting constraint will be. However, one needs to be careful here not to use a measure that is too strong as to make the constraint unsatisfiable. In particular, one should not insist on using optimal plans here. For instance, consider the following scenario:

- initially the robot is in room $r1$ and she needs to deliver the object that she's holding to location $l2$ in room $r2$;
- the rooms $r1$ and $r2$ are connected by door d;
- the robot's hand needs to be empty in order to close a door;

[5]One should look at this axiom in conjunction with possibly other constraints such as the executability of the plan σ.

- at door d, the nearest place that the robot can put down an object is the location $l1$ in room $r1$.

It can be seen that for a scenario like this, there is no plan σ that will achieve the goal of having the object at location $l2$ and satisfy at the same time the constraint (15) with *Bminimal* replaced by a formula corresponding to optimal plans: as soon as the robot enters room $r2$, she needs to go back to room $r1$ because the optimal plan for closing door d requires the robot to put down the object that she's holding at the nearest location which happens to be in room $r1$. This is the reason that we mentioned earlier in section 3 when we argued for the need of suboptimal plans.

Notice that axiom (15) requires the robot to close a door as soon as possible the last time she goes through it, but leaves open what to do with the door at other times. To be more stringent, a user may require that every time the robot opens a door in order to go through it, she has to close it as soon as possible: for any door d, and any action α in the plan, if $\neg closed(d)$ becomes true after α is performed, then as soon as $through(d)$ becomes true, the next segment of the plan should be a plan for $closed(d)$:

$$\begin{aligned}&(\forall d,s,a).[do(a,s) \sqsubseteq S_{end} \wedge \neg H(closed(d),s) \wedge\\ &\quad H(closed(d),do(a,s))] \supset\\ &\qquad (\exists s')[do(a,s) \sqsubset s' \sqsubseteq S_{end} \wedge H(through(d),s')] \wedge\\ &\qquad (\forall s')\{do(a,s) \sqsubset s' \sqsubseteq S_{end} \wedge H(through(d),s') \supset\\ &\qquad\quad (\exists s'')[s' \sqsubset s'' \sqsubseteq S_{end} \wedge\\ &\qquad\quad Bminimal(closed(d),s',s'')]\} \qquad (16)\end{aligned}$$

Constraint (16) is a generalization of that in [2] except that we do not require the robot to go through a door as soon as she opens it. To do that, we would have to axiomatize why the robot needs to go through the door in the first place, which is, while something worth doing, beyond the scope of the present paper.[6]

Of course, a formal specification of desirable behaviours is only the first step. More difficult questions concern formal properties of the specifications and methods to generate plans that satisfy them. This, we believe, is where the advantage of working under the situation calculus, in particular basic action theories, comes in. For instance, for the robot domain, it can be shown that if there is a plan that satisfies (14), then there is also a plan that satisfies (15).

[6]For instance, if the robot is trying to carry an object to another room, then after she opens the door, the next goal should be going through the door to the other room *while carrying the desired object.*

6 Related work

To the best of our knowledge, the most closely related work is that by Foulser, Li, and Yang [5]. They consider what they call the plan merging problem: given that "knowledge is available about what operators can be merged, and for each set of these operators, what the merged operators are... finding and analyzing methods for computing the optimal plan" (page 156, [5]). On the one hand, our work complements that in [5] in that we built into our definitions what operators can be "merged." For instance, according to B-minimality, sets of mergeable operators are identified with linear segments of the given plan, and the merged operator is the empty sequence (or the idle "wait" operator). On the other hand, our emphasis is different than that in [5]. Given a plan σ, while Foulser, Li, and Yang would look for the optimal, e.g. B-minimal "subplan" of σ, we are only concerned with obtaining a B-minimal subplan, i.e., for us, one B-minimal plan is as good as any other ones.

Our work is also related to that by Ambite and Knoblock [1] who propose a planning strategy that divides the planning process into two steps: initial plan generation, and plan refinement. While much of [1] is about domain specific rules that would transform a given plan into a "better" one, we are mainly interested in domain independent measures here. It would be an interesting future research topic to see how useful a domain independent notion of plan quality such as one of those studied here would be to a planning strategy such as Ambite and Knoblock's.

7 Concluding remarks

We have argued for the need of a formal notion of plan quality other than the usual notion of optimal (shortest) plans. In particular, we have proposed three local, tractable, and increasingly restrictive measures of plan quality, and studied their properties. We have provided a logical formalization for them in the situation calculus, show their relationships with regression and STRIPS triangle tables, and explore their possible applications in the formal specifications of robots' behaviours.

This work is part of our project on using the situation calculus to formalize control knowledge in planning, and is motivated primarily by the robot navigating example discussed above. There are many possible directions for future work. One of them is to use the situation calculus to provide a full declarative specification of the robot's desirable behaviors in the navigating domain.

Acknowledgements

I would like to thank the two anonymous KR'98 reviewers for their useful comments about an earlier version of this paper. This research was supported in part by the Research Grants Council of Hong Kong under Competitive Earmarked Research Grant HKUST6091/97E.

References

[1] J. L. Ambite and C. A. Knoblock. Planning by rewriting: efficiently generating high-quality plans. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97), AAAI Press, Menlo Park, CA.*, pages 706–713, 1997.

[2] F. Bacchus and F. Kabanza. Planning for temporally extended goals. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI-96), AAAI Press, Menlo Park, CA.*, pages 1215–1222, 1996.

[3] R. E. Fikes, P. E. Hart, and N. J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3(4):251–288, 1972.

[4] R. E. Fikes and N. J. Nilsson. STRIPS: A new approach to theorem proving in problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[5] D. E. Foulser, M. Li, and Q. Yang. Theory and algorithms for plan merging. *Artificial Intelligence*, 57:143–181, 1992.

[6] F. Lin. Applications of the situation calculus to formalizing control and strategic information: the prolog cut operator. In *Proceedings of the Fifteen International Joint Conference on Artificial Intelligence (IJCAI-97), IJCAI Inc. Distributed by Morgan Kaufmann, San Mateo, CA.*, pages 1412–1418, 1997.

[7] F. Lin. An ordering on subgoals for planning. *Annals of Mathematics and Artificial Intelligence*, 21:321–342, 1997.

[8] F. Lin and R. Reiter. State constraints revisited. *Journal of Logic and Computation, Special Issue on Actions and Processes*, 4(5):655–678, 1994.

[9] J. McCarthy and P. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, Edinburgh, 1969.

[10] E. P. Pednault. Synthesizing plans that contain actions with context-dependent effects. *Computational Intelligence*, 4:356–372, 1988.

[11] E. P. Pednault. ADL: Exploring the middle ground between STRIPS and the situation calculus. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning (KR'89)*, pages 324–332. Morgan Kaufmann Publishers, Inc., 1989.

[12] R. Reiter. The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 418–420. Academic Press, San Diego, CA, 1991.

[13] R. Reiter. On specifying database updates. *Journal of Logic Programming*, 25(1):53–91, 1995.

[14] R. Reiter. *Knowledge in Action: Logical Foundations for Describing and Implementing Dynamic Systems.* In preparation, 1997.

[15] H. Turner. A logic of universal causation. *Artificial Intelligence*, 1998. To appear.

[16] D. Weld and O. Etzioni. The first law of robotics (a call to arms). In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94), AAAI Press, Menlo Park, CA.*, pages 1042–1047, 1994.

Representing Granularity and Vagueness

Modal Semantics for Knowledge Bases Dealing with Vague Concepts

Brandon Bennett
Division of Artificial Intelligence
School of Computer Studies
University of Leeds, Leeds LS2 9JT, England
`brandon@scs.leeds.ac.uk`

Abstract

The paper investigates the characterisation of vague concepts within the framework of modal logic. This work builds on the *supervaluation* approach of Fine and exploits the idea of a *precisification space*. A simple language is presented with two modalities: a necessity operator and an operator 'it is unequivocal that' which is used to articulate the logic of vagueness. Both these operators obey the schemas of the logic $S5$. I show how this language can be used to represent logical properties of vague predicates which have a variety of possible precise interpretations. I consider the use within KR systems of number of different entailment relations that can be specified for this language. Certain vague predicates (such as 'tall') may be indefinite even when there is no ambiguity in meaning. These can be accounted for by means of a three-valued logic, incorporating a definiteness operator. I also show the relationship between observable quantities (such as height) and vague predicates (such as 'tall') can be represented *via* axioms involving precise comparative relations (such as 'taller'). I consider how Williamson's 'logic of clarity' can be combined with the semantics for unequivocality and how the clarity operator can be related to observables.

1 INTRODUCTION

Natural language descriptions are pervaded by the use of vague concepts of a variety of different kinds. Perhaps the most often cited cases are predicates such as 'tall', where it is not clear exactly how high a person or object should be to count as being tall. Here the vagueness is associated with the problem of reconciling a qualitative concept with a continuous range of possible measurements. In more subtle cases it may not even be clear what objective facts are at stake. (Consider the meaning of 'disaster' in the sentences 'The eruption of Krakatoa was a disaster' and 'The best man's speech was a complete disaster'.) Often the use of a seemingly simple concept may involve different kinds of vagueness at the same time. For instance, in a sentence such as 'This frog is green' the word 'green' is vague not only in that there may be borderline cases of green (a yellowy green or a dark browny green) but also because the frog may have a yellow belly, black eyes and a pink tongue. Hence, 'green' in some circumstances means predominantly green rather than green all over. Although the different types of vagueness share certain properties, it is implausible that all these phenomena could be accounted for by a completely uniform logical analysis.

Vagueness is often thought of as a deficiency of natural language: it is imagined that if vague concepts were replaced by clearly defined concepts the resulting precise language would be superior to existing natural languages. Against this view it may be argued that vagueness actually increases the utility of language because it enables one to make true statements without having to use the extremely cumbersome descriptions that would be needed to make them completely precise.

In constructing AI knowledge bases, researchers have typically attempted to eliminate vagueness from concepts by precise definitions. However, vagueness poses an extremely serious threat to this enterprise. Since ordinary language concepts are vague it is very easy for contradictions to occur because of slight differences in the intended meaning of different occurrences of predicates in a set of axioms. In the construction of large axiomatised ontologies (Guha and Lenat 1990) this problem is especially acute. I believe that a logical treatment of vagueness can be particularly useful in preventing inconsistency arising in this way. Similar problems also occur when one attempts to combine information from two or more different databases (Lehmann and Cohn 1994).

In this paper I shall show that by a suitable logical treatment of vagueness it is possible to safely combine data and axioms where the meanings of predicates may vary from instance to instance. Moreover, I shall show that it is possible to do this without completely enfeebling inference capabilities. Whether or not vagueness is a defect or a asset, the ability to manipulate vague concepts would certainly be very useful in a wide range of KR&R systems.

1.1 RELATION OF THE PRESENT APPROACH TO PREVIOUS WORK

In computer science the approaches to vagueness that have received the most attention are those based on ***multi-valued*** logics (Lukasiewicz and Tarski 1930). ***Fuzzy*** logic (Zadeh 1965, Zadeh 1975, Goguen 1969, Dubois and Prade 1988) can be regarded as a generalised multi-valued logic. An extensive discussions of the pros and cons of fuzzy logic can be found in (Elkan 1993). In accord with Elkan, I suggest that fuzzy logic may be an appropriate formalism for modelling the relation between continuous valued observables and the meanings of vague qualitative predicates; however, it is not suitable as a formalism for carrying out logical reasoning. This is primarily because propositional operators whose values are determined purely in terms of the fuzzy truth values of their arguments cannot take account of either logical or domain-specific constraints holding among the argument propositions.

Another approach to vagueness was given by Fine (1975) in terms of ***supervaluations***. The idea is that vague expressions are associated not with a single extension but with a set of different possible extensions. This can be modelled by means of a Kripke-style possible worlds semantics (Kripke 1959). In a recent book by Williamson (1994), it is argued that vagueness should be regarded as a epistemic phenomenon and proposes a modal treatment of vagueness rather different from that of Fine. This paper also takes a model-theoretic approach incorporating and extending a number of previous ideas. However, whereas previous models were proposed mainly in support of philosophical thesis about vagueness, I shall be concerned with the management of vagueness within AI knowledge bases. Consequently, my concern is to specify a formal language within which logical dependencies between vague expressions can be represented and which provides a complete inference mechanism.

1.2 VAGUENESS, UNCERTAINTY AND GENERALITY

I distinguish sharply between vagueness and uncertainty.[1] Although they are both associated with lack of precision of information and may both be regarded as properties of propositions, the two properties are complementary rather than parallel. The more uncertain a proposition is, the less likely it is to be true; but the more vague it is the more likely it is to be in some sense true. In other words by stating information in terms of vague concepts we may be more certain about our claims. Another common misunderstanding is that generality and disjunctive information are a form of vagueness or of uncertainly. For instance one might (wrongly) regard the statement 'I am in my twenties' as more vague than 'I am 29' but although the first sentence is less specific it is not at all vague (and both are — or were — equally certain): being vague (but just as certain) I would say 'I am approaching 30'. Generality and disjunction are similar to vagueness in that they are ways of making certain (or at least more certain) claims regarding situations about which we do not know all the precise details.

In this paper I shall assume that the state of the world is completely determinate in terms of certain precise *objective measurements* that can always obtained for any given objects in any given situation. These determine the truth of all precise propositions and also indirectly determine the validity of inferences which involve vague concepts. The content of this assumption will be made precise by means of a formal (possible worlds) semantics. I also assume that vagueness is confined to the predicates of the language. This means that there are no vague objects[2] or nominal terms and the logical operators are also taken to be completely definite.

[1]The subject of uncertainty has also received much attention from AI researchers (see e.g. (Shafer and Pearl 1990)).

[2]The logic of vague objects is likely to be very different from that of vague predicates. A theory of vague spatial objects can be found in (Cohn and Gotts 1996).

2 PRECISIFICATION SPACES AND SUPERVALUATIONS

The supervaluation theory of vagueness proposed by Fine (1975) is based on the idea that expressions of a language which includes vague expressions can be assigned different semantic values according to different possible ways in which the vague expressions can be interpreted.[3] Each such interpretation is called a *precisification*. The simplest model of this is to identify each precisification with a classical two-valued truth-assignment to the atomic propositions of the language. Precisifications are then treated just like possible worlds in a Kripke semantics. Fine's semantics has been elaborated and clarified by Pinkal (1995).

2.1 MODELS FOR NECESSITY AND UNEQUIVOCALITY OPERATORS

Fine's analysis of vagueness assumes we can specify a set of *admissible* precisifications. These are associated with valuations of atomic propositions which are taken to reasonably accord with natural intuitions about the meaning of vague concepts. Of course, from a purely formal point of view one cannot distinguish between reasonable and unreasonable valuations but Fine takes admissibility as a primitive notion restricting the range of possible models. Whilst this is perfectly coherent it presents methodological problems in that it obscures the relationship between valuations and analytic constraints on predicate meanings. Such constraints are assumed to be dealt with before the model theoretic analysis begins. By contrast I impose no constraints on precisifications — they are associated with arbitrary valuations. This enables necessary constraints to be modelled axiomatically in the logic rather than treated as restrictions imposed from outside. Thus, I identify validity with truth which is guaranteed in arbitrary valuations, not just in those that are admissible. Within this more general framework the concept of an 'admissible' precisification can easily be introduced as being one satisfying some necessary theory, Θ: we say that 'ϕ is true for all admissible precisifications' corresponds to the condition that $\Theta \rightarrow \phi$ is true for arbitrary valuations.

A formal language for knowledge representation will typically be used to specify theoretical as well as factual relationships. In my analysis of vagueness, the difference between these two types of proposition will be important and it will be useful to explicitly distinguish them both syntactically and semantically. To do this I employ the modal logic $S5$ (see e.g. (Hughes and Cresswell 1969)). This enables theoretical information to be represented by formulae of the form $\Box\phi$ (factual formulae will not contain modal operators). In order to articulate the logic of vagueness I introduce into the language the modality $\mathbf{U}$ meaning 'unequivocally' and its dual $\mathbf{S}$ (where $\mathbf{S}\phi \equiv \neg\mathbf{U}\neg\phi$) meaning 'in some sense'. Thus, for example, $\mathbf{U}P \rightarrow \mathbf{S}Q$ asserts that if P is unequivocally true (i.e. true under any reasonable interpretation) then Q is in some sense true (i.e. true under some reasonable interpretation). This extended language will be called $S5\mathbf{U}$.

As a model for an intensional logic of vagueness, within which the necessity modality can used to specify analytic truths, I employ a Kripke semantics in which each 'world' is indexed by a pair $\langle w, p\rangle$, w being a member of a set of possible worlds W and p being a member of a set of precisifications P. I shall refer to the semantic entity referred to by a pair $\langle w, p\rangle$ as a *point*. Each point is associated with a classical valuation of formulae which do not contain modal operators.

An $S5\mathbf{U}$ model is a structure $\langle W, P, I, \mathcal{R}, \delta\rangle$, where I is a set of individual constants, $\mathcal{R}$ is a set of relation symbols and δ is a function from $W \times P \times \mathcal{R}$ to tuples of elements of I. I write $\langle w, p\rangle \Vdash \phi$ to mean that ϕ is true at $\langle w, p\rangle$. For an atomic proposition $R(x_1, \ldots, x_n)$ I specify that

$$\langle w, p\rangle \Vdash R(x_1, \ldots, x_n) \text{ iff } \langle x_1, \ldots, x_n\rangle \in \delta(w, p, R)$$

The truth-conditions of the Boolean truth functions and quantifiers can then be specified in the usual way:

$$\begin{array}{lll} \langle w,p\rangle \Vdash (\alpha \wedge \beta) & \text{iff} & \langle w,p\rangle \Vdash \alpha \text{ and } \langle w,p\rangle \Vdash \beta \\ \langle w,p\rangle \Vdash (\alpha \vee \beta) & \text{iff} & \langle w,p\rangle \Vdash \alpha \text{ or } \langle w,p\rangle \Vdash \beta \\ \langle w,p\rangle \Vdash \neg\phi & \text{iff} & \langle w,p\rangle \nVdash \phi \\ \langle w,p\rangle \Vdash \forall x\phi(x) & \text{iff} & (\forall i \in I)\{\langle w,p\rangle \Vdash \phi(i)\} \end{array}$$

The semantics for quantification is based on the simplifying assumption that the domain of quantification (I) is constant for every world/precisification point $\langle w, p\rangle$. This is probably not realistic. A more adequate theory would need a more complex semantics with variable domains and would need to address problems of identifying counterpart individuals at different worlds. Such considerations are beyond the scope of the present paper.

The truth-conditions for the necessity an unequivocality operators are as follows:

$$\langle w,p\rangle \Vdash \Box\phi \quad \text{iff} \quad (\forall u \in W)\{\langle u,p\rangle \Vdash \phi\}$$

$$\langle w,p\rangle \Vdash \mathbf{U}\phi \quad \text{iff} \quad (\forall q \in P)\{\langle w,q\rangle \Vdash \phi\}$$

[3]The term 'supervaluation' appears to originate with van Frassen, who used this kind of semantic structure to model the phenomenon of *presupposition* which occurs in natural languages (van Frassen 1968, van Frassen 1969).

A formula ϕ is valid in this semantics if it is forced at every world/precisification in every model. I write $\models_{S5\mathbf{U}} \phi$.

The proof theory of this language can be specified in at least two different ways. One method is to fist translate formulae of $S5\mathbf{U}$ into purely 1st-order formulae. By reifying propositions and the indices of the semantic points and propositions, a condition $\langle w, p\rangle \Vdash \phi$ can be represented by a formula $\mathsf{holds}(w, p, \phi)$. The semantics conditions for the connectives in $S5\mathbf{U}$ can easily be specified as axioms constraining the holds relation. Given that I am assuming a constant domain of quantification for every point $\langle w, p\rangle$, quantifiers occurring in ϕ can be moved outwards to operate on the holds relation. The meaning of the modal operators can be expressed directly by quantification over the index variables. For unmodalised formulae the indices will be fixed constants w_0 and p_0 denoting the the 'actual' world/precisification.

Alternatively, we may want to formulae the proof theory within the object language of $S5\mathbf{U}$. To do this we add to a proof procedure for 1st-order logic the usual $S5$ schemata and the rule of necessitation for each of the two operators $\Box$ and $\mathbf{U}$. For completeness we will also need additional schemata expressing logical interactions between the two different modalities. The most obvious of these is the commutation schema,

$$\Box\mathbf{U}\phi \leftrightarrow \mathbf{U}\Box\phi\,. \qquad \textbf{(Com)}$$

Segerberg (1973) gave a '2-dimensional' modal logic which is very similar, although rather more expressive than my logic $S5\mathbf{U}$. This includes an axiom equivalent to

$$\Box\mathbf{U}(\Box\alpha \vee \mathbf{U}\beta) \leftrightarrow (\Box\mathbf{U}\alpha \vee \Box\mathbf{U}\beta) \qquad \textbf{(Orth)}$$

which enforces a property pertaining to the orthogonal nature of the two modalities. The completeness proof given by Segerberg strongly suggests that these schemata provide a complete axiomatisation of $S5\mathbf{U}$. However, Segerberg's system contains additional schemata involving other modal operators and it is possible that these induce further properties of $\Box$ and $\mathbf{U}$, which must be axiomatised for completeness.

By means of well-known theorems of $S5$ together with the commutation schema it is easy to show that there is a finite number of 'modalities' in $S5\mathbf{U}$ — i.e. a finite number of logically distinct strings of the symbols $\neg$, $\Box$, $\Diamond$, $\mathbf{U}$, and $\mathbf{S}$. For example, it is fairly easy to show that

$$\Box\mathbf{S}\Box\phi \leftrightarrow \mathbf{S}\Box\phi\,.$$

All the distinct modalities which do not include negation are shown in Figure 1 (arrows indicate entailment). Each has a negated counterpart, so that the total number of modalities is 22. $S5$ also has the property that any formula can be systematically reduced to an equivalent formula within which no modal operator occurs within the scope of an other modal operator. This reduction allows all $S5$ formulae to be transformed into a relatively simple normal form (MCNF — see (Hughes and Cresswell 1969)) and forms the basis of decision procedure for $S5$ theoremhoood. It may be possible to establish a similar though considerably more complex normal form for $S5\mathbf{U}$. This would be very useful for automated theorem proving.

$$\begin{tikzcd}
 & & \mathbf{S}\Box \arrow[dr] & & \\
 & \Box \arrow[ur] \arrow[r] \arrow[dr] & \Box\mathbf{S} \arrow[r] & \mathbf{S} \arrow[dr] & \\
\mathbf{U}\Box \arrow[ur] \arrow[rr] \arrow[dr] & & \text{none} \arrow[ur] \arrow[rr] \arrow[dr] & & \mathbf{S}\Diamond \\
 & \mathbf{U} \arrow[ur] \arrow[r] \arrow[dr] & \mathbf{U}\Diamond \arrow[r] & \Diamond \arrow[ur] & \\
 & & \Diamond\mathbf{U} \arrow[ur] & &
\end{tikzcd}$$

Figure 1: Distinct modalities in S5U

2.2 CONTRAST WITH MULTI-VALUED APPROACHES

The $S5\mathbf{U}$ semantics means that every classical theorem is unequivocally true even if it contains vague predicates. Thus

$$\models_{S5\mathbf{U}} \mathbf{U}(\mathsf{tall(John)} \vee \neg\mathsf{tall(John)})$$

and

$$\models_{S5\mathbf{U}} \mathbf{U}\neg(\mathsf{tall(John)} \wedge \neg\mathsf{tall(John)})\,.$$

This is because each precisification/world is associated with a purely classical valuation, which makes every classical theorem true. By contrast, classical theorems involving vague predicates are generally not valid in many-valued or truth-functional logics, where the truth-value of a conjunction or disjunction is determined solely by the semantic value of the argument propositions. In such a logic, if tall(John) is not definitely true or false then it will have some indefinite truth value and so will ¬tall(John). It may also be that thin(John) is also indefinite and has the same semantic value as tall(John). Hence, in such a case the tautology $\neg(\mathsf{tall(John)} \wedge \neg\mathsf{tall(John)})$ must have the same truth value as the contingent proposition $\neg(\mathsf{thin(John)} \wedge \neg\mathsf{tall(John)})$.

2.3 PRECISIFICATIONS AND PREDICATE SENSES

A precisification space is an abstract structure exhibiting the ways in which a language can be made precise in different ways. In the case of a real learnable language this space must be determined by the different possible interpretations of specific expressions. Since I am assuming that object names and logical operators are perfectly precise, the possible interpretations of any expression must be determines by the possible interpretations of its constituent predicates.

Consider the natural language predicate 'x is a murderer'. Although some necessary and sufficient properties of a murderer are uncontroversial, other properties are open to debate. Thus there is a wide variety of possible precise definitions of murderer. For instance one definition would be that: a murderer is a human being who has intentionally killed another human being. However, some people (e.g. certain strict vegetarians) might contend that a human being who intentionally kills any kind of animal is also a murderer.

To be more specific about the relationship between 'killer' and 'murderer' we could stipulate that if someone is a murderer in any sense then they are unequivocally a killer:

$$\Box\forall x[\ (\mathbf{S}\,\mathsf{murderer}(x)) \rightarrow \mathbf{U}\,\mathsf{killer}(x)]\ .$$

but in the opposite direction we might have the weaker implication

$$\Box\forall x[\ \mathsf{killer}(x) \rightarrow (\mathbf{S}\ \mathsf{murderer}(x))]\ .$$

If we then give axioms specifying the meaning of these concepts there will be certain analytic facts that we are sure should be true whatever precise sense they are given. For example:

$$\Box\,\mathbf{U}(\mathsf{killer}(x) \leftrightarrow \exists y[\mathsf{killed}(x,y)])$$

$$\Box\,\mathbf{U}(\mathsf{murderer}(x) \rightarrow \exists y[\mathsf{human}(y) \wedge \mathsf{killed}(x,y)])$$

Other axioms will only apply to some but not all interpretations of vague predicates:

$$\Box\,\mathbf{S}\forall x[\mathsf{murderer}(x) \leftrightarrow \exists y[\mathsf{animal}(y) \wedge \mathsf{killed}(x,y)]]\ .$$

Sometimes to specify properties of vague predicates it may be useful to introduce artificially sharpened versions of these predicates. Thus, if we wanted to use 'murderer1' in a sense which is sharper than the vague predicate 'murderer', we might use the following axiom:

$$\Box\forall x[\mathsf{murderer1}(x) \rightarrow (\mathbf{S}\,\mathsf{murderer}(x))] \wedge$$

$$\Box\forall x[(\mathbf{U}\,\mathsf{murdererer}(x)) \rightarrow \mathsf{murderer1}(x)]$$

and define murderer1 by

$$\Box\,\mathbf{U}\forall x[\mathsf{murderer1}(x) \leftrightarrow \exists y[\mathsf{human}(y) \wedge \mathsf{killed}(x,y)]]\ .$$

Equivocal predicates may include those that are obviously ambiguous, having two or more quite separate meanings rather than a cluster of similar meanings. One way to distinguish certain ambiguous concepts is to say that there is no case to which it applies in every sense. Thus we could write

$$\neg\Diamond\exists x[\mathbf{U}\ \mathsf{nut}(x)]$$

to say that there is no possible situation in which there is something that is unequivocally a 'nut' (nothing is both a nut of the edible variety and of the mechanical variety).

As well as vague and ambiguous predicates, most knowledge bases will contain a large number of precise predicates, whose meaning is not in question. The sharpness of certain predicates can be attested by axioms such as

$$\Box\forall x[(\mathbf{S}\,\mathsf{integer}(x)) \rightarrow (\mathbf{U}\,\mathsf{integer}(x))]\ .$$

It should be clear form the various examples given in this section that the $S5\mathbf{U}$ language provides an expressive representation for describing the logical properties of vague predicates. Especially those whose vagueness consists in them having a cluster of related meanings.

2.4 VALIDITY DEFINED USING THE MODAL DEFINITELY OPERATOR

Within a precisification semantics such as $S5\mathbf{U}$ it is possible to define the relation of entailment among vague propositions in a number of different ways. Philosophical examination of this semantics has often sought to demonstrate that some particular definition is correct. However, from the point of view of describing possible inferences within a database containing vague information, it is not necessary to adopt a single definition of entailment. Many kinds of entailment can be defined in terms of the unambiguous property of validity and each of these tells us something different about the logical relationship between propositions.

Some of the more useful possible definitions of entailment are the following:

- $\phi_1, \ldots, \phi_n \models_{\mathrm{local}} \psi$ iff $\models_{S5U}$ $\mathbf{U}((\phi_1 \wedge \ldots \wedge \phi_n) \rightarrow \psi)$
- $\phi_1, \ldots, \phi_n \models_{\mathrm{global}} \psi$ iff $\models_{S5U}$ $(\mathbf{U}\phi_1 \wedge \ldots \wedge \mathbf{U}\phi_n) \rightarrow \mathbf{U}\psi$
- $\phi_1, \ldots, \phi_n \models_{\mathrm{arguable}} \psi$ iff $\models_{S5U}$ $\mathbf{S}((\phi_1 \wedge \ldots \wedge \phi_n) \rightarrow \psi)$
- $\phi_1, \ldots, \phi_n \models_{\mathrm{reliable}} \psi$ iff $\models_{S5U}$ $(\mathbf{S}\phi_1 \wedge \ldots \wedge \mathbf{S}\phi_n) \rightarrow \mathbf{U}\psi$
- $\phi_1, \ldots, \phi_n \models_{\mathrm{s-reliable}} \psi$ iff $\models_{S5U}$ $(\mathbf{S}\phi_1 \wedge \ldots \wedge \mathbf{S}\phi_n) \rightarrow \mathbf{S}\psi$
- $\phi_1, \ldots, \phi_n \models_{\mathrm{co-reliable}} \psi$ iff $\models_{S5U}$ $\mathbf{S}(\phi_1 \wedge \ldots \wedge \phi_n) \rightarrow \mathbf{U}\psi$
- $\phi_1, \ldots, \phi_n \models_{\mathrm{co-s-reliable}} \psi$ iff $\models_{S5U}$ $\mathbf{S}(\phi_1 \wedge \ldots \wedge \phi_n) \rightarrow \mathbf{S}\psi$

Philosophers (Fine 1975, Williamson 1994) have been most concerned with the first two definitions of validity. So-called 'local' validity means that under-any interpretation where the premisses are true the conclusion is also true. However, because $\mathbf{U}$ satisfies the rule of necessitation and the T schema, $\models_{S5U}$ $\mathbf{U}\phi$ holds just in case $\models_{S5U}$ ϕ; so, for formulae not containing modalities, local validity is equivalent to classical validity.

To avoid this reduction of entailment in vague languages to classical validity, it has been argued that an appropriate characterisation of valid entailment in vague languages is that if the premisses are unequivocally true the conclusion is also unequivocally true. This has been called 'global' validity and is strictly weaker than local validity (the former is derivable from the latter by the K schema).

What I call 'arguable' validity is perhaps the weakest useful entailment relation. A sequent is arguably valid if the corresponding implication is valid under some interpretation. If we move the $\mathbf{S}$ operator inwards in the schematic theorem corresponding to arguable entailment we get $((\mathbf{U}\,\phi_1 \wedge \ldots \wedge \mathbf{U}\,\phi_n) \rightarrow \mathbf{S}\,\psi)$. Thus to show that a sequent is arguably valid we only need to show that if all the premisses are 'unequivocally' true then the conclusion is 'in some sense' true. This form of entailment is suitable for query applications, where we are interested in whether a vague query might possibly be a consequence of some vague data. Ideally a procedure which tests for arguable validity would also give further information about what senses of the query and database facts make the entailment valid. The user could then assess the reliability of the inference.

'Reliable' entailment on the other hand is very strong. It makes valid only those arguments such that however the premisses are interpreted, the conclusion is unequivocally true. This notion of entailment might be useful in certain safety critical applications where we want to take some action only if an entailment must hold whatever interpretations are given to vague concepts in the premisses and conclusion.

The remaining three entailment relations are weaker than 'reliable' but are similar in that we only assume that the premisses are 'in some sense' true. For 's-reliable' entailment we only require that some sense of the conclusion is implied whatever sense is given to the premisses. In 'co-reliable' and 'co-s-reliable' we assume that, although all the premisses are only true 'in some sense', they are all interpreted consistently. Thus the $\mathbf{S}$ operator acts on the conjunction of premisses rather than each premiss separately.

Notice that, when we are concerned with 'reliable' and 's-reliable' entailment, two separate database facts α and β are not equivalent to the conjunction $\alpha \wedge \beta$. This is because we assume that uses of a predicate within a single formula are consistent; whereas, two distinct formulae which share one or more vague predicates may be true for different interpretations of these predicates.

2.5 THE SORITES ARGUMENT

I now consider the famous *sorites* paradox which is perhaps the best known example of the way that vagueness seems to defy logical analysis. Detailed accounts may be found in (Black 1970, Burns 1991, Williamson 1994). The original example of this paradox concerns the question of how many grains of sand constitute a 'heap' ('sorites' is Greek for heap) but I shall consider the directly analogous case of the bald man (which will be of more relevance to many academics).

Assume we have a theory in which arithmetic addition is defined. The following argument is then classically valid:

1. $\forall x[(\mathsf{num\_hairs}(x) = 0) \rightarrow \mathsf{bald}(x)]$
2. $\forall x \forall y[(\mathsf{bald}(x) \wedge (\mathsf{num\_hairs}(y) = \mathsf{num\_hairs}(x) + 1)) \rightarrow \mathsf{bald}(y)]$
3. $\therefore$ $\forall x[(\mathsf{num\_hairs}(x) = 10{,}000{,}000) \rightarrow \mathsf{bald}(x)]$

Precisification semantics enables one to avoid this paradox because in border-line cases we can say that a person is 'bald' under some interpretations of this concept but not under others. One can then weaken the second premiss to say: if an n-haired man is bald under all reasonable interpretations of 'bald' then an $(n+1)$-haired man is bald under at least one reasonable interpretation of 'bald':

$$\forall x \forall y[(\mathbf{U}\,\mathsf{bald}(x) \wedge (\mathsf{num\_hairs}(y) = \mathsf{num\_hairs}(x)+1)) \\ \rightarrow \mathbf{S}\,\mathsf{bald}(y)]$$

Under this interpretation, the argument is not valid according to the semantics for U given above.

This analysis is perhaps not a complete solution of the paradox. One might argue that if someone with n hairs is bald 'in some sense' then someone with $n+1$ hairs must be bald in some (slightly weaker) sense. So the paradox re-emerges with respect to the predicate '... is bald in some sense'. It may be argued that fuzzy logic gives a better account of sorites-style paradoxes because it allows the degree to which someone is bald to increase monotonically as hairs are lost (see e.g. (Goguen 1969). However, other problems with fuzzy logic were noted above (section 2.2).

3 ANALYTIC AND OBSERVATIONAL VAGUENESS

The $S5\mathbf{U}$ semantics given in section 2.1 assumes that every precisification assigns classical truth values (true or false) to every atomic predicate. This model gives a good account of vague concepts that correspond to a cluster of precise concepts, each having definite truth conditions. However, there are also cases where there seems to be no ambiguity in the meaning of the word but nevertheless there is indeterminacy as to when the concept should be applied. For instance, the predicate $\mathsf{tall}(x)$ does not seem to have a variety of senses, although there is no specific height which qualifies a person as being tall. In judging the truth of a proposition such as $\mathsf{tall}(\mathsf{John})$ it seems perfectly reasonable to say in certain cases that its truth is indefinite. Of course different judges might still quibble about what was an indefinite case.

In $S5\mathbf{U}$ we can say that a proposition is indefinite in that it is neither true in every sense nor false in every sense: $\neg\mathbf{U}\,\phi \wedge \mathbf{S}\phi$. However we cannot say that a proposition is indefinite in every sense. This motivates the construction of a three-valued precisification semantics.

3.1 THREE-VALUED PRECISIFICATION SPACES

In a three-valued precisification model (Fine 1975, Pinkal 1995), a precisification is associated with a function mapping each atomic predicate and each tuple of arguments to a value in the set $\{\mathbf{t}, \mathbf{f}, \mathbf{i}\}$ (true, false and indefinite). A precisification which only assigns the values $\mathbf{t}$ and $\mathbf{f}$ is called *complete*. If a precisification p assigns the value $\mathbf{t}$ whenever q assigns $\mathbf{t}$ and $\mathbf{f}$ whenever q assigns $\mathbf{f}$, then we write $p \geq q$ and say that p *extends* q.

Fine suggests that one should constrain the space of precisifications $\mathcal{P}$ by what he calls the Resolution Principle (RP) requiring that whenever an atom indefinite at a precisification there is an extension of that precisification which makes it true and another that makes it false. In formal terms this means that for any precisification $p \in \mathcal{P}$ and any atomic proposition A, if $[A]_p = \mathbf{i}$, then there must exist precisifications $q, r \in \mathcal{P}$ such that $q \geq p$, $r \geq p$, $[A]_q = \mathbf{t}$ and $[A]_r = \mathbf{f}$. RP means that all maximal elements of $\mathcal{P}$ are complete. A further suggested constraint is that there should be a minimal precisification p_0 of which every world is an extension.

If RP is dropped and we consider models which have arbitrary sets of valuations we get a very general semantics within which a wide variety of operators can be defined. Such models will not be explored here but they may well prove useful in modelling aspects of vagueness, lack of knowledge or uncertainty.

Fine investigates a number of ways in which the semantics of complex propositions should be determined by a 3-valued precisification space. The simplest proposal is that the truth value of a complex proposition at a precisification p is determined by the truth values assigned according to 2-valued classical logic to all complete extensions of p. Specifically, ϕ is true at p if it is true at all complete extensions of p; ϕ is false at p if it is false at all complete extensions of p; otherwise ϕ is indefinite. This is very plausible semantics for propositions involving only classical connectives. Fine also suggests that a definiteness operator $\mathbf{D}$ can be added to the language. He gives two possible semantic specifications for this operator. One is that the truth of a formula $\mathbf{D}\phi$ depends only on the value of ϕ at the minimal (i.e. most indefinite) world p_0:[4]

$$p \Vdash \mathbf{D}\phi \quad \text{iff} \quad p_0 \Vdash \phi$$

According to this specification, the truth of $\mathbf{D}\phi$ is con-

[4]Under this definition, $\mathbf{D}$ is essentially the same as Pinkal's '$\boxdot$' operator.

stant over all precisifications.

Fine also suggests a more complex semantics, within which a proposition may be definite at some precisifications and not at others. This involves treating the **D** operator as a 3-valued truth functional operator whose truth-table is given in Table 1. The table also shows how the Boolean connectives can be treated as truth functional except in the case of a conjunction $\alpha \wedge \beta$ of two indefinite propositions. Here the truth value of the conjunction is not a simple truth-function of the values of the conjuncts because, even though α and β may be indefinite, it may be impossible for them to be true together (e.g. α might be 'John is tall' and β 'John is not tall'). To make this precise, we stipulate that when $[\alpha]_p = \mathbf{i}$ and $[\beta]_p = \mathbf{i}$, then: $[\alpha \wedge \beta]_p = \mathbf{i}$ iff there is some precisification $q \geq p$ (q is an extension of p) such that $[\alpha]_q = \mathbf{t}$ and $[\beta]_q = \mathbf{t}$; otherwise $[\alpha \wedge \beta]_p = \mathbf{f}$.

ϕ	$\neg\phi$
t	f
i	i
f	t

$\alpha \wedge \beta$	t	i	f
t	t	i	f
i	i	i/f	f
f	f	f	f

ϕ	$\mathbf{D}\phi$
t	t
i	f
f	f

Table 1: Truth tables for $\neg$ and $\wedge$ and **D**.

It should be noted that the **D** operator is semantically very different from the **U** operator. $\mathbf{D}\phi$ is true at a world/precisification just in case ϕ is true at that world/precisification, whereas $\mathbf{U}\phi$ is true at a world if ϕ is true at that world for every precisification. At the semantic level it is easy to combine both operators into a single language. We just replace the 2-valued precisification space of $S5\mathbf{U}$ with the 3-valued space. In the combined language we can say things like $\mathbf{S}(\neg\mathbf{D}\,\mathsf{tall}(\mathsf{John}) \wedge \neg\mathbf{D}\neg\mathsf{tall}(\mathsf{John}))$, meaning that 'in some sense John is neither definitely tall nor definitely not tall'.

The object-level proof theory of this language is somewhat problematic. However, the semantics can easily be described in classical 1st-order logic in a similar fashion to the method suggested in section 2.1. This provides an indirect way of testing validity.

4 OBSERVABLES AND VAGUENESS

The three-valued precisification semantics allows one to model predicates which are vague even when their interpretation is unequivocal. However, it does not provide any means for specifying the way in which the justification for applying a vague predicate depends on the state of the world. In this section I examine how the truth of a vague qualitative concept may depend on the value of some continuously variable observable parameter. To do this I consider a language which contains, as well as predicates (which may be more or less vague) functions corresponding to the values of observable properties.

4.1 OBJECTIVE MEASURES AND POSSIBLE WORLDS

It is plausible that the meanings of vague concepts depend to some extent on objective measurements. To formalise these I introduce a class of entities which will be called *observables*. These can be regarded as types of measurement which can be applied to objects. Each observable has a fixed arity, which is the number of objects to which it applies. To make the model theory simpler I shall assume that we have only monadic and dyadic observables. Examples of observables are **height** and **weight** (monadic) and **distance** (dyadic).

I consider each possible world to be associated with a complete assignment of real values to every n-ary observable for each n-tuple of arguments. Thus each world will assign a height and weight to each object and a distance to each pair of objects. This semantics for precise observables can readily be combined with a precisification semantics for vague predicates. A model for a vague language with observable functions is then a structure

$$\mathcal{M}_{VLO} = \langle W, P, I, \mathcal{R}, \delta, O_1, \sigma_1, O_2, \sigma_2, \rangle ,$$

where: $\langle W, P, I, \mathcal{R}, \delta\rangle$ is a precisification model of the kind described in section 2.1; O_1 is a set of names of monadic observables; σ_1 is a function from $W \times O \times I$ to real numbers; O_2 is a set of names of dyadic observables and σ_2 is a function from $W \times O \times I \times I$ to real numbers.

Note that unlike the valuation of relations, the valuation of observable functions does not depend on the precisification. Thus these functions are objective (non-vague) properties of the world.[5] The possible values of observables will normally be constrained by a theory Θ_{obs}, which may be regarded as a formalisation of physical laws. Thus the observable **distance** might be axiomatised to be a Euclidean metric.

[5]One could easily add vague functions to the language also. These would have values that could vary between different precisifications for the same world.

4.2 PREDICATE MEANINGS

In a setting where there is no uncertainty about the state of the word and no controversy regarding the sense of a concept, the justification as to whether a tuple of objects satisfies some predicate will appeal to the values of observables evaluated with respect to those objects. To model these justification conditions one may axiomatise precise predicates by specifying their necessary connections to observables.

For vague predicates the situation is more complex. Some vague predicates are uncontroversially linked with a single observable. In such a case the predicate may still be vague because the cut-off value for the predicate may be debatable. An example of this is the predicate tall. We can assume that this is determined purely by the observable **height**. (Of course some argument might arise in comparing heights when one person has a stoop or curly hair. However this kind of vagueness is best modelled with the **U** operator and can be separated from that which occurs even when the height measurement is uncontroversial.) Many vague predicates will depend upon more than one observable. For example whether a person is 'big' may depend on both their height and weight. A tall person might be described as big even if lighter than a shorter person. On the other hand if the tall person were only moderately tall but extremely thin one would be reluctant to call him 'big'.

Another problem is that concepts such as tall are relative to a class. Thus a tall child is not as tall as a tall woman. This phenomenon is undoubtedly a significant feature of natural vague concepts however it will not be tackled in the current paper.

4.3 COMPARATIVES AND THE PROPERTIES OF VAGUE CONCEPTS

Despite the fact that the applicability of a vague predicate such as tall is indeterminate, comparative relations which seem to be derived from these predicates may be completely determined by observables. For instance the relation taller could be defined in terms of **height** as follows:

$$\Box(\forall x \forall y[\mathsf{taller}(x,y) \leftrightarrow (\mathbf{height}(x) > \mathbf{height}(y))])$$

Here the truth of the mathematical comparative relation '>' is determined directly by a model of the form of $\mathcal{M}_{VLO}$.

It is also clear that willingness to ascribe the predicate tall to a person increases the greater their height; so, if any reasonable judge calls one person tall, then she must regard all taller people as tall:

$$\mathbf{U}\,\Box(\forall x \forall y[(\mathsf{tall}(x) \wedge \mathsf{taller}(y,x)) \rightarrow \mathsf{tall}(y)])$$

Axioms involving comparatives provide a straightforward and very general way of constraining the meanings of vague concepts.

5 THE LOGIC OF CLARITY AND THE 'DISTANCE' BETWEEN WORLDS

Williamson (1994) proposes that vagueness should be described by means of a propositional operator, $\mathbb{C}$, where $\mathbb{C}\,\phi$ is read 'It is clear that ϕ'. The meaning of this operator is defined relative to a possible worlds semantics in which there is a metric over the set of possible worlds giving a 'distance' between any two worlds. We then say that a proposition $\mathbb{C}\,\phi$ is true at a world w (i.e. ϕ is *clearly* true at w) just in case ϕ is true at all worlds within a certain distance of w. This distance is taken as providing a *margin of error* which ensures that ϕ is clearly true since it is true not only at the actual world w but at all worlds which differ from w by some small amount. In this section I look at the relationship between Williamsons's logic of clarity, precisification spaces and observables.

It seems reasonable to suppose that the distance between possible worlds is determined by observables and that the margin of error required for an atomic proposition to be 'clearly' true will vary according to the meaning of its predicate. Thus each predicate can be associated with a metrical function of observables and a margin. Together these constitute a relevant *measure* for that predicate. It is implausible that the precise specification of this measure is a fixed property of the predicate. Rather these will vary according to a spectrum of ways in which the predicate could be interpreted. Moreover, one might expect that each distinct predicate would be associated with its own space of measures. However, Williamson considers an idealisation of this picture in which, within a given model, the certainty of all predicates is determined by a single metric and margin on the possible worlds.

The fact that we do not need separate measures for each predicate may seem odd; but the notion of a metric is very general and can encompass distance within a multi-dimensional space of possible variations of atomic predicates.

Williamson considers specifying the semantics of $\mathbb{C}$ in terms of *fixed margin models*. Such a model is a quadruple $\langle W, d, \alpha, [\,]\rangle$, where W is a set, d a metric[6] on W, α a non-negative real number, and $[\,]$ is a mapping of formulas to subsets of W such that for all formulas ϕ, ψ:

- $[\neg\phi] = W - [\phi]$
- $[\phi \wedge \psi] = [\phi] \cap [\psi]$
- $[\mathbb{C}\,\phi] = \{w \in W : (\forall x \in W)[\,(d(w,x) \leq \alpha) \rightarrow x \in [\phi]\,]\}$

Under these conditions it can be shown that $\mathbb{C}$ behaves exactly as the $\Box$ operator of the modal logic $S4$. However, Williamson argues that this operator is to strong in that it obeys the schema $\Box\,\phi \leftrightarrow \Box\,\Box\,\phi$. This would rule out the possibility of second-order vagueness as exemplified by a claim such as 'John is clearly tall but not clearly clearly tall'.[7] In order to allow for non-redundant iteration of $\mathbb{C}$, this operator can be interpreted in terms of a *variable margin model* where

$$[\mathbb{C}\,\phi] = \{w \in W : (\exists\delta > \alpha)(\forall x \in W)[\,(d(w,x) \leq \delta) \rightarrow x \in [\phi]\,]\}\ .$$

If validity is identified with truth in all such models then $\mathbb{C}$ is constrained by exactly the same schemata as the $\Box$ operator of the modal logic KT. Since the schemata K and T are intuitively correct properties of any plausible clarity operator and any refinement of the idealised semantics could only weaken the logical properties of $\mathbb{C}$, Williams argues that KT provides an adequate specification of the logic of clarity.

The semantics of clarity is orthogonal to the idea of precisification in that to evaluate $\mathbb{C}\,\phi$ at a some world/precisification, we look at the valuations for a predicate at a neighbouring worlds with respect to the same precisification. To take account of this model-theoretically, one could generalise the notion of precisification to include a measure on possible worlds as well as valuation on predicates. Since models include arbitrary precisifications, every possible combination of predicate valuations and measures will correspond to a point in the precisification space. The reason that measures are considered part of precisifications is that I take them to be part of the sense ascribed to a predicate. Hence if we combine $\mathbb{C}$ with the unequivocality operator **U**, the **U** operator asserts that a proposition is true for every assignment *and* every measure (not just every assignment for a fixed measure).

By adding measures to a precisification model (either the simple $S5$**U** model or the 3-valued version) one can construct a semantics within which **U** and $\mathbb{C}$ (and in the 3-valued case also **D**) are definable. This we can say things like 'The eruption of Krakatoa was clearly a disaster in every sense'. This means that for every precisification the proposition 'The eruption of Krakatoa was a disaster' is true for every possible world which is within the required margin of similarity of the actual world (the appropriate margin being dictated by the precisification).

The logic of $\mathbb{C}$ can also be related to the values of observables. Since the distance measure between possible worlds is determined by observables, in a language including observable functions it is possible to constrain the behaviour of the clarity operator to take account of this dependence. One way to do this is in terms of comparatives. Analogously to the example given in section 4.3 one could state the following axiom ensuring that if someone is clearly tall according to some precisification then every taller person is also clearly tall:

$$\mathbf{U}\,\Box(\forall x \forall y[(\mathbb{C}\,\mathsf{tall}(x) \wedge \mathsf{taller}(y,x)) \rightarrow \mathbb{C}\,\mathsf{tall}(y)])$$

6 CONCLUSION

This paper has drawn on previous work in philosophical analysis of vagueness in order to construct formalisms which may prove useful for handling vague concepts within AI knowledge bases. Although the semantic models I have presented are not completely original, they seem to be little known to AI researchers. This is perhaps because they have hitherto been developed principally for the exposition of philosophical arguments rather than for practical purposes. Consequently, in order to specify useful representation languages for dealing with vague information, certain modifications are required.

By the seemingly trivial generalisation of precisification spaces to include all arbitrary valuations of relations rather than just 'admissible' ones I have provided a logically neutral model structure within which analytic (i.e. necessary) properties of both vague and sharp predicates can easily be stated. I have suggested that philosophical debates concerning the correct definition of entailment within a vague language are irrelevant to KR. Moreover each of variety of possible entailment relations can yield useful information about possible inferences among propositions involving vague concepts.

I have distinguished the vagueness that occurs where a predicate has a cluster of different senses from the vagueness associated with the lack of a clear cut off point for a qualitative predicate with respect to continuously variable objective measurement. I believe that these different kinds of vagueness require quite

[6] i.e. it satisfies the usual definition of a metric.

[7] I am not really sure how much sense this makes.

distinct model theoretic treatment but can nevertheless be incorporated within a single, albeit rather complex, semantics. Further elaboration of the details of this semantics is the subject of ongoing work. My approach contrasts with that of other treatments (such as fuzzy logic) where different types of vagueness (and sometimes also uncertainty) are treated uniformly. I think that my analysis suggests that such an assimilation is unlikely to give a satisfactory of the various aspects of vagueness.

Vagueness is an extremely subtle phenomenon which is very hard to get hold of within a logical framework. Consequently there are likely to be many deficiencies in the semantical models I have proposed. Nevertheless I believe that my semantic analysis is in many respects correct and can provide a framework which will enable AI systems to take account of certain important aspects of vagueness. I also believe that a satisfactory account of vagueness is essential to the solution of many crucial problems in AI. Vagueness is a fundamental property of natural languages and consequently must be taken seriously in any KR system that attempts to manipulate natural concepts.

Acknowledgements

This work was supported by the EPSRC under grant GR/K65041.

I would like to thank professor A.G. Cohn for helpful discussions on the subject matter of this paper.

References

Black, M.: 1970, Reasoning with loose concepts, *Margins of Precision*, Cornell University Press, Ithaca, chapter 1, pp. 1–13.

Burns, L. C.: 1991, *Vagueness: An Investigation Into Natural Languages And The Sorites Paradox*, Vol. 4 of *Reason and Argument*, Kluwer.

Cohn, A. G. and Gotts, N. M.: 1996, A mereological approach to representing spatial vagueness, *in* J. D. L C Aiello and S. Shapiro (eds), *Principles of Knowledge Representation and Reasoning, Proc. 5th Conference*, Morgan Kaufmann, pp. 230–241.

Dubois, D. and Prade, H.: 1988, An introduction to possibilistic and fuzzy logics, *in* P.Smets, E.H.Mamdani, D.Dubois and H.Prade (eds), *Non-Standard Logics for Automated Reasoning*, Academic Press, London.

Elkan, C.: 1993, The paradoxical success of fuzzy logic, *Proceedings of the National Conference on Artificial Intelligence (AAAI-93)*, pp. 698–703.

Fine, K.: 1975, Vagueness, truth and logic, *Synthèse* **30**, 263–300.

Goguen, J.: 1969, The logic of inexact concepts, *Synthese* **19**, 325–373.

Guha, R. V. and Lenat, D. B.: 1990, Cyc: a mid-term report, *AI Magazine* **11**(3), 32–59.

Hughes, G. E. and Cresswell, M. J.: 1969, *An Introduction to Modal Logic*, Methuen, London.

Kripke, S.: 1959, A completeness theorem in modal logic, *Journal of Symbolic Logic* **24**, 1–14.

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

Łukasiewicz, J. and Tarski, A.: 1930, Untersuchungen über den aussagenkalkül, *Comptes rendus des séances de la société des sciences et des lettres de Varsovie* **3**(23), 1–21. Reprinted in A. Tarski, *Logic, Semantics, Metamathematics*, 1956, Clarendon Press, trans. J.H. Woodger.

Pinkal, M.: 1995, *Logic and Lexicon: the semantics of the indefinite*, Vol. 56 of *Studies in linguistics and philosophy*, Kluwer, Dordrecht. translated by Geoffrey Simmons.

Segerberg, K.: 1973, Two-dimensional modal logic, *Journal of Philosophical Logic* **2**, 77–96.

Shafer, G. and Pearl, J. (eds): 1990, *Readings in Uncertain Reasoning*, Morgan Kaufmann, Palo Alto.

van Frassen, B. C.: 1968, Presupposition, implication and self-reference, *Journal of Philosophy* **65**, 136–52.

van Frassen, B. C.: 1969, Presupposition, supervaluations and free logic, *in* K. Lambert (ed.), *The Logical Way of Doing Things*, Yale University Press, New Haven, chapter 4, pp. 67–91.

Williamson, T.: 1994, *Vagueness*, The problems of philosophy, Routledge, London.

Zadeh, L.: 1965, Fuzzy sets, *Information and Control* **8**, 338–353.

Zadeh, L. A.: 1975, Fuzzy logic and approximate reasoning, *Synthese* **30**, 407–428.

A Theory of Granularity and its Application to Problems of Polysemy and Underspecification of Meaning

Inderjeet Mani
The MITRE Corporation, W640
11493 Sunset Hills Road
Reston, Virginia 22090, USA
imani@mitre.org

Abstract

Communication using natural language is remarkably efficient, by allowing reuse (through the use of generative devices) of a finite vocabulary to describe a potentially infinite set of situations. This vocabulary reuse contributes to words having many related senses (polysemy). Further, meanings can be relatively vague or precise; in other words, varying in their degree of specification of meaning. I suggest that these problems can be addressed by developing a knowledge representation which makes explicit the notion of granularity. As the grain size changes, we may fold certain distinctions, or split meanings more finely. In this paper, I formalize a theory of granularity and demonstrate how it can be applied to problems of meaning representation. Such a theory requires a world model which provides a rich sortal differentiation of entities based on the distinctions made by natural language, including the representation of meronymic structure and reification. Granularity will be represented in terms of structural operations defined as abstractions. I illustrate how this applies to problems of polysemy and vagueness in nominalizations, where splitting and folding of meanings are particularly evident.

1 INTRODUCTION

Natural language provides us humans with a very powerful means of describing the world as we experience it. Communication using natural language is remarkably efficient, by allowing reuse (through the use of generative devices) of a finite vocabulary to describe a potentially infinite set of situations. This vocabulary reuse contributes to words having many related senses (polysemy). Further, meanings can be relatively vague or precise; in other words, varying in their degree of specification of meaning. If every aspect of a word's meaning were made explicit in everyday communication, communication would become very inefficient. These properties of natural language communication, polysemy and vagueness/underspecification, pose challenges for knowledge representation, in particular for lexical semantics, where the meanings of words have to be represented.

Lexical ambiguity in natural language can be viewed as falling into two disjoint categories: homonymy and polysemy. A lexically ambiguous word is polysemous iff it has multiple semantically related meanings. If a word has meanings which are semantically unrelated, those unrelated meanings are called homonymies. For an examples of polysemy, consider (1a-b). In (1a), *the bottle* refers to bottle as a container; in (1b) it refers to the contents.

(1a) Mary broke the bottle.

(1b) Mary finished the bottle.

For another example of polysemy, consider (2a), in which the nominalization is ambiguous at least between a process reading (2b) and a result reading of an object created (2c). (The term "nominalization" as used here means a noun or noun-headed constituent - such as a noun phrase - which can denote something event-like.) The result reading can be replaced by an ordinary count noun where one exists, but a count noun can't be used in the process reading.

(2a) The painting lasted three hours.

(2b) The painting/*picture lasted three hours, because they had to stop work that evening.

(2c) The painting/picture lasted three hours, because it was so flimsy.

In both (1) and (2), the shift in meaning involves some inferential path which relates the default meaning to the meaning expected by the predicate (e.g., "break"/"finish" in (1)). Clearly, the inferential paths involved in these examples encode some notion of perspective shift.

It is worth noting that perspective shifts can also result in information being lost. This is seen in each of (3a-c) (adapted from [Pustejovsky 95]), where the nominalization appears to denote both a process and a result object:

(3a) The house's construction was finished in two months.

(3b) John's construction of the roof frame for the house was done yesterday.

(3c) John's construction of the roof frame for the house was done yesterday, and even though it took seven hours to get done, when it was done, it looked real good.

Such readings are underspecified with respect to whether we are dealing with a process or a result object.

There are numerous other cases where perspective shifts across basic ontological categories can occur, as when we shift from describing a *meeting* in terms of (say) a time interval to a description in terms of specific instants, as in (4a) (from [Pianesi and Varzi 96]), or from a description of a *pencil point* as a point to its being a surface, as in (4b) (from [Asher and Vieu 95]):

(4a) That's how they met: *at a certain point*, John asked the waiter to invite her at his table; *the next moment* she was sitting in front of him.

(4b) The point of this pencil is actually an irregular surface with several peaks.

It is clear, then, that the examples in (1-4) all involve a shift in perspective. Shifts in perspective, in turn, reflect efficiencies in reasoning processes. Collectively, these examples suggest that it may be helpful to view efficiencies in language as a reflection of efficiencies in reasoning processes. Such efficiencies are very closely related to Hobbs' notion of "granularity": [Hobbs 85] suggests that in the course of reasoning we conceptualize the world at different levels of granularity, and that in a particular reasoning process we distinguish only those things that are relevant to that process. As [Hobbs 85] puts it, "our knowledge consists of a global theory together with a large number of relatively simple, idealized, grain-dependent, local theories, interrelated by articulation axioms." In any given situation, a granularity is determined, allowing the local theory to be selected. Since the local theory has a smaller, less complex ontology than one which would obtain if all predicates were always relevant, there is more hope of exploiting computationally tractable reasoning processes.

In what follows, I will show how problems of polysemy and underspecification can be addressed by a theory of reasoning which makes explicit the notion of granularity. The overall idea is that as the grain size changes, we may fold certain distinctions, or split meanings more finely. This idea complements existing work, providing a foundation for looking at problems in natural language semantics in terms of a theory of reasoning processes. In current approaches to semantics for natural language processing, cases of meaning shift like (1-2) are handled by some form of type coercion, involving, e.g., lexical rules defined over typed feature structures [Briscoe and Copestake 91], or [Pustejovsky 95] building it into the combination operations (e.g., function application) used to assemble phrases and sentences out of words. Thus, a system would have "container/contents" rules. The account I propose here does not seek to supplant these traditional rules; rather, I suggest that such rules are part of a more general inferential process involving granularity shifts. However, I go beyond previous approaches to address cases like (3) as well in terms of granularity shifts. Thus, in applying a theory of granularity to problems of lexical semantics, I will characterize the representational requirements in more abstract ways, providing a relatively problem-independent classification of the inferential procedures needed.

For reasons of space, I confine myself here to illustration of this idea with respect to problems of polysemy and underspecification in nominalizations, such as (2) and (3). This allows us to focus on the representation of event structure in the ontology, which allows us to make relatively precise certain key distinctions we have described informally, such as "event", "process", "result", "process or result", etc. The extension to (4a) will follow from this; the extension to (4b) requires some analysis of mereotopology (cf. [Asher and Vieu 95]), which we leave to future work.

In order to analyze the above phenomena, we have to first represent some of the distinctions which natural languages make of the world, which we take up in the next section.

2 ONTOLOGICAL DISTINCTIONS FOR NATURAL LANGUAGES

Following [Bach 86], let us define a join relation $\cup$ in R over the **entities** in D, so that if $q1$ and $q2$ are entities, $q1 \cup q2$ is also an entity. (For convenience, terms given a definition in our ontology will be boldfaced, except when it's obvious that it's the technical sense that's intended.) The inclusion ordering relation $\preceq$, which defines a join semilattice over the set of entities, is:

Definition 1 (inclusion) $\forall(q1, q2)\ q1 \preceq q2 \equiv q1 \cup q2 = q2$

Among the sorts of entities are time **periods** and **individuals**. Instants will approximated as very short time periods (of course, alternative approaches, e.g., treating instants as primitive and defining intervals in terms of them, [Dowty 79], are clearly possible). Both **objects** and **eventualities** are subsorts of **individuals**.

Eventualities are further subdivided into the Vendler [Vendler 67] sorts of **states**, **processes** (activities), **accomplishments**, and **achievements** (the latter three subsorts will be called **events**). As a notational device, we will use sortal variables $q1$, $q2$ to stand for entities, $t1$, $t2$ for time periods, x, y for individuals, $v1$, $v2$ for eventualities, e for events, p for processes, s for states. **Processes** differ from other **events** in that if a process p (like John's walking) occurs in period t, it must also occur for every proper sub-period of t (this definition excludes gapped processes). **Accomplishments** are events e of which the primitive predicate $culminate(e, t)$ can hold. (Intuitively, an event culminates in t if at the end of t the event has finished.) **Achievements** are instantaneous accomplishments (instants here are very short periods). Both accomplishments and achievements are defined so that if they occur over a time period t they may not occur in any other time period $t1$ such that $t1 \preceq t$.

In addition to representing **individuals** and time **periods** as subsorts of **entities**, we will also explicitly represent **sets** as another subsort of **entities**. This is needed, at the very least, to represent **propositions**. A proposition p is usually conceived of as the set of possible worlds where p is true. Here, a proposition will be represented as denoting a set of times, namely, the set of times in which the content of the proposition is true. As such, propositions can be viewed as abstract individuals representing sets, and we therefore include **abstract individuals** as a further primitive category. Facts (found in factive expressions such as *John admits that Mary came*, or *John admits to the fact that Mary came*) are simply true propositions (following [Zucchi 93]).

Associated with a set S will be a special kind of abstract individual called an **individual correlate** $^{\cap}S$ of the set, which is essentially an individual standing for that set. The concept of an abstract individual is relatively well established (e.g., [Chierchia 82], [Link 83], [Hwang and Schubert 93], [Krifka 87], etc.). Summation operators have of course been discussed elsewhere in the literature, e.g., [Link 83], [Krifka 87], [Landman 89]. For reasons of space, I forego a comparison here with these earlier works (see [Mani 97b] for more discussion).

I now introduce one specialized inclusion relation:

Definition 2 (aspectual inclusion)
$\forall(v1, v2)\ v1 \preceq_W v2 \equiv v1 \preceq v2$ *and* $v1$ *is an aspectually salient part of* $v2$

By introducing the term "aspectually salient", I am recognizing that the inclusion relation might be dense, in that an event may be broken down into an infinite number of parts, corresponding to the dense linear ordering of the real numbers. In some instances, we only want to consider subparts which relate to aspectual substructure of the event. Thus for example, an accomplishment event which is an event of building a house can be broken down into two aspectually salient parts, a preparatory process and a resulting state. The resulting state overlaps with the event, and can of course extend beyond it.

Now, I will define a ternary relation first described by [Pustejovsky 95]:

Definition 3 (exhaustive ordered part-of (*eop*))
$\forall(v1, v2, v3)\ eop(v1, v2, v3) \equiv$
$v1 \preceq_W v3 \wedge v2 \preceq_W v3 \wedge event\_meets(v1, v2)$
$\wedge v3 = v1 \cup v2$
$\wedge(\forall v4\ \textit{if}\ v4 \preceq_W v3,\ \textit{then}\ v4 \preceq v1 \vee v4 \preceq v2)$

Thus for example, an accomplishment event $a1$ which is an event of building a house will have a preparatory process $p1$ and a resulting state $s1$ related by $eop(p1, s1, a1)$. For another example (from [Pustejovsky 95]) the event $e1$ of killing can be viewed as having a process (call it **killing-act**) $p1$ and a resulting state (call it **being-dead**) $s1$, related to $e1$ by $eop(p1, s1, e1)$.

The *eop* definition relies on the following auxiliary definitions (here **occurs** - for events - and **holds** -for states - are treated as primitives):

Definition 4 (happens) $\forall(v, t)\ happens(v, t) \equiv$
(i) v is an **event** *and t is the maximal interval over which* $occurs(v, t)$ *or (ii) v is a* **state** *and t is the maximal interval over which* $holds(v, t)$

Definition 5 (event_meets) $\forall(v1, v2, t1, t2)$
$event\_meets(v1, v2) \equiv$
$happens(v1, t1) \wedge happens(v2, t2)\ \wedge meets(t1, t2)$

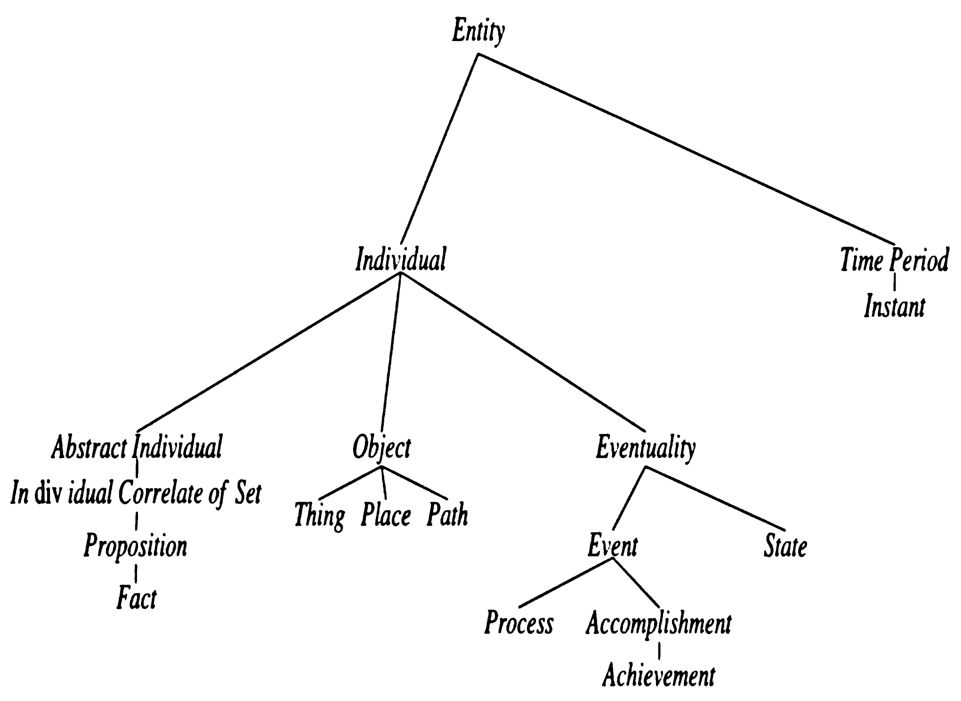


Figure 1: Ontological Distinctions

Here, for any $t1$, $t2$, $meets(t1, t2)$ iff $t1$ ends before $t2$ starts, with no other period in between. In somewhat more precise terms (derived in part from [Allen 81], [Allen 83], [Allen 84]):

Definition 6 (meets) $\forall(t1, t2)$ $meets(t1, t2) \equiv \exists(t1e, t2s)$ *such that* $t1$ *ends at* $t1e$ *and* $t2$ *starts at* $t2s$ *and* $t1e$ *precedes* $t2s$ *and* $\forall t3$, ($t3$ *precedes* $t1e$) $\vee$ ($t2s$ *precedes* $t3$)

Finally, we introduce an informal definition which says that x is the result object associated with an eventuality if it takes the patient role in the eventuality and it is around during the time of the result state associated with the eventuality:

Definition 7 (eventuality result-object (ero)) $\forall(v, x)$ $ero(v, x) \equiv$
$\exists(s, p, t2, t1)$ *such that* $happens(v, t2) \wedge eop(p, s, v) \wedge patient(v)(x) \wedge holds(s, t1) \wedge exists\_at(x, t1)$

The sorts we have enumerated here are ordered by the sortal ordering relation $\subseteq$. All the sorts, along with $\subseteq$, are shown in Figure 1.

3 ANALYSIS USING TYPE-SHIFTING OPERATORS

Now that we have represented certain key distinctions in our ontology, we are in a position to analyze the linguistic examples we began with. As mentioned earlier, cases of meaning shift are treated in contemporary approaches as involving some form of type coercion. Prior to characterizing these in terms of granularity, however, I will first provide a type-shifting analysis based on the ontological distinctions introduced in the previous section.

Before launching into this account, let me state certain assumptions. I assume that the meanings of natural language sentences can be represented by logical expressions, each of which has a denotation in a world model. These logical expressions representing the meanings of sentences will be called "logical forms". I will have to assume that the reader has a basic familiarity with the lambda calculus [Church 41], as well as the use of categorial grammars and type-theoretic representations for the semantics of natural languages (see [Dowty 79]). Finally, it may be helpful if the reader has some familiarity with Davidsonian semantics [Davidson 67], [Davidson 69], [Parsons 90], though it is not essential.

Now, let's return to (2) above. I will assume that there is a basic reading of a nominalization corresponding to a $\overline{N}$ constituent (i.e., a noun-headed constituent which combines with a determiner like *the*), as denoting a set of **events**. Thus, the noun *construction* has the logical form (5a), and the $\overline{N}$ *construction* has the logical form (5b), where the arguments are filled (i.e., "saturated") from context.

(5a) $\lambda e\ \lambda x\ \lambda y\ [constructing(e) \wedge agent(e)(y) \wedge patient(e)(x)]$

(5b) $\lambda e \exists (x, y)\ [constructing(e) \wedge agent(e)(y) \wedge patient(e)(x)]$

Now, in many situations, (5b) is a perfectly satisfactory representation of the meaning of (the $\overline{N}$) *construction.* Consider what happens when the grain size shifts, causing the meanings to be split more finely. We can define a type shifting operator which shifts the meaning from an event to a process:

(6) $\chi \equiv \lambda Q \lambda p \exists (e, s)[Q(e) \wedge eop(p, s, e)]$

Applying this operator to (5b), we get the logical form for the process reading (which we illustrated in (2b)), where the nominalization denotes a set of **processes**:

(7) $\lambda p \exists (e, x, y)[constructing(e) \wedge agent(e)(y) \wedge patient(e)(x) \wedge eop(p, s, e)]$

Likewise, we have a type-shifting operator θ:

(8) $\theta \equiv \lambda Q \lambda z \exists e[Q(e) \wedge ero(e, z)]$

θ extracts the result **object** meaning (as was illustrated in (2c)) when applied to (5b), and this is shown in (9):

(9) $\lambda z \exists (e, x, y)[constructing(e) \wedge agent(e)(y) \wedge patient(e)(x) \wedge ero(e, z)]$

Note that although there are three possible polysemies: **process**, result **object**, and result **state**, we never get a three-way polysemy in a single use: we either have process/result object (*the construction*), or we have process/result state (*the destruction*). Given a $\overline{N}$ expression like *destruction*, ω extracts the result state meaning when applied to (5b).

(10) $\omega \equiv \lambda Q \lambda s \exists (e, p)[Q(e) \wedge eop(p, s, e)]$

Next, we turn to the cases of underspecified readings of "process or result" in (3). Here, we define an underspecification operator:

(11) $\mu \equiv \lambda Q \lambda e[e = {}^{\cap}\chi Q \cup {}^{\cap}\theta Q]$

The result of applying μ to (5b) is (12a), which when expanded to (12b), reveals that it denotes a set of **individual correlates** of the set of **processes** and result **objects** denoted by the process and result object readings, respectively:

(12a) $\mu \lambda e \exists (x, y)[constructing(e) \wedge agent(e)(y) \wedge patient(e)(x)]$

(12b) $\lambda e1[e1 = \lambda p \exists (e, s, x, y)[constructing(e) \wedge agent(e)(y) \wedge patient(e)(x) \wedge eop(p, s, e)] \cup \lambda z \exists (e, x, y)[constructing(e) \wedge agent(e)(y) \wedge patient(e)(x) \wedge ero(e, z)]]$

Likewise, there is another underspecification operation which folds **processes** and result **states**:

(13) $\vartheta \equiv \lambda Q \lambda e[e = {}^{\cap}\chi Q \cup {}^{\cap}\omega Q]$

Finally, we have an underspecification operator which folds **events** and **propositions**:

(14) $\xi \equiv \lambda Q \lambda e[e = {}^{\cap}theQ \cup {}^{\cap}compQ]$

where $the \equiv \lambda P \iota e[P(e)]$ maps one-place predicates to individuals, ι is the usual iota operator, and $comp \equiv \lambda P \exists e[P(e)]$ maps one place predicates to times.

It is worth noting that underspecification over individuals (such as individual processes or results) is but one of a variety of different forms of underspecification that have been studied in the literature, including underspecified scope representations [Reyle 93]. The approach to underspecification described here applies only to underspecification over individuals.

4 GRANULARITY

Now, I will try to make good on my promise to characterize type shifting operators in terms of more general inferential processes involving changes in granularity. In order to do so, however, I must first flesh out a theory of granularity. Once I have done that, I will return in the next section to the linguistic phenomena.

As mentioned earlier, [Hobbs 85] suggests that in the course of reasoning we conceptualize the world at different levels of granularity, and that in a particular reasoning process we distinguish only those things that are relevant to that process, making other things indistinguishable for all practical purposes. In particular, Hobbs defines an indistinguishability relation such that two domain entities x and y are indistinguishable iff for all relevant predicates p, $p(x)$ is true iff $p(y)$ is true. The particular level of granularity determines which predicates are relevant. Thus, a theory in which temperatures are distinguished to the nearest degree, can, via an indistinguishability relation, be mapped to one in which temperatures are distinguished to the nearest 10 degrees (e.g., *in the fifties*). In any given situation, a granularity is determined, allowing the local theory to be selected. When the grain size shifts, certain "articulation axioms" are used to map to another local theory.

In general, a mapping which induces a change in granularity can be considered a special case of an **abstraction** [Giunchiglia and Walsh 92], which we will consider here to be a total, surjective function from formal languages $\mathcal{L}1$ to $\mathcal{L}2$. I restrict myself here to abstractions where the source and target language are the same language $\mathcal{L}$ (though there will be lots of applications - outside the scope of this paper - where the languages are different formal languages). I will write $\mathcal{F}_{\gamma}^{\Leftrightarrow}$ to mean an abstraction which uses a function γ to map a source logical form to an equivalent (denotationally identical) target logical form, both drawn from the same language $\mathcal{L}$. Likewise, let $\mathcal{F}_{\gamma}^{\Rightarrow}$ (or $\mathcal{F}_{\gamma}^{\Leftarrow}$) mean an abstraction which maps a source logical form to an entailed (or entailing) target logical form. As a notational convention, I will drop the entailment superscript if it isn't specified. The notions of equality of abstractions (based on extensional equality of operators), identity, composition and inverse are similar to those defined in [Giunchiglia and Walsh 92] (cf. pp. 370-71), for reasons of space, these are excluded here.

As [Giunchiglia and Walsh 92] also point out, Hobbs' notion of granularity can be represented in terms of a **granularity abstraction** which we define here as:

Definition 8 (granularity abstraction) *An abstraction $\mathcal{F}_{\gamma}^{\Rightarrow}$ is a granularity abstraction iff*

(i) $\mathcal{F}_{\gamma}^{\Rightarrow}$ maps individual constants in $\mathcal{L}$ to their equivalence class under the indistinguishability relation $\sim$ in $\mathcal{L}$. (Thus, for any individual x in $\mathcal{L}$, $\mathcal{F}_{\gamma}^{\Rightarrow}(x) = \kappa(x)$ where $\kappa(x) = \{y$ such that $x \sim y\}$.)
and
(ii) $\mathcal{F}_{\gamma}^{\Rightarrow}$ maps everything else, including the predicates in $\mathcal{L}$, to itself.

Note that granularity abstractions may be particularly prone to lose information, since elements which are indistinguishable (given the context) are collapsed.

For problems of natural language semantics (which [Giunchiglia and Walsh 92] do not address), it is important to establish whether applying an abstraction to a logical form which is then combined with another logical form is the same as applying the abstraction to the result of the combination of the two logical forms. If it is the same, then the abstraction preserves compositionality. I will call such an abstraction **endocentric**, defined as follows:

Definition 9 (endocentric abstraction)
Given logical forms α, β in $\mathcal{L}$, and a binary combination operator $*$ which forms new logical forms, an abstraction $\mathcal{F}_{\gamma}$ is endocentric iff
(i) $\mathcal{F}_{\gamma}(\alpha * \beta) = \mathcal{F}_{\gamma}(\alpha) * \beta$
or
(ii) $\mathcal{F}_{\gamma}(\alpha * \beta) = \mathcal{F}_{\gamma}(\beta) * \alpha$

Among the abstractions we will introduce and exploit are **meronymic abstractions**:

Definition 10 (meronymic abstraction) *An abstraction $\mathcal{F}_{\gamma}$ is a meronymic abstraction iff $\forall\alpha$ in $\mathcal{L}$,*
(i) α denotes an entity $q1$, and $\mathcal{F}_{\gamma}(\alpha)$ denotes an entity $q2$ such that $q2 \preceq q1$
or
(ii) α denotes a set of entities $Q1$, and $\mathcal{F}_{\gamma}(\alpha)$ denotes a set of entities $Q2$ such that $\forall q2$ in $Q2$ $\exists q1$ in $Q1$ such that $q2 \preceq q1$

Similarly, we can represent **sortal abstractions**, whose definition is identical to Definition 10, except that $\subseteq$ is used instead of $\preceq$. To distinguish meronymic from sortal abstractions, and to indicate their direction, we will introduce another subscript, e.g., ${}_{\preceq}\mathcal{F}_{\gamma}$.

It is worth noting that a meronymic abstraction which maps an individual to its equivalence class under the indistinguishability relation $\sim_1$, defined below, is a granularity abstraction.

Definition 11 (indistinguishability under $^{\cap}$)
$\forall(x,y) x \sim_1 y \equiv x \preceq {}^{\cap}\{x,y\}$ *and* $y \preceq {}^{\cap}\{x,y\}$

The nature of the entailment varies with the abstraction. For granularity abstractions, we have upward entailment $\Rightarrow$; for sortal abstractions ${}_{\subseteq}\mathcal{F}$, we have downward entailment $\Leftarrow$. In other cases, the entailment, if any, depends on the specifics of the logical form and the abstraction operator.

Finally, we introduce a notion of **grain size**:

Definition 12 (relative grain size) $\forall(\alpha,\beta)$ *in $\mathcal{L}$, α is* **finer-grained** *than β iff there is an abstraction ${}_{R}\mathcal{F}_{\gamma}$ such that ${}_{R}\mathcal{F}_{\gamma}(\alpha) = \beta$ and either*
(i) $R = \preceq$
or
(ii) $R = \subseteq$

5 TYPE SHIFTING OPERATORS AND ABSTRACTIONS

We now return to the analyses of Section 3, but this time with a characterization of meaning shifts in terms of abstractions. We can define a family of meronymic abstractions ${}_{\preceq}\mathcal{F}_{\chi}$, ${}_{\preceq}\mathcal{F}_{\theta}$, and ${}_{\preceq}\mathcal{F}_{\omega}$, where the operators χ, θ, and ω are as defined earlier in (6), (8) and (10). None of these abstractions are endocentric, as the associated operator takes an event-selecting predicate and returns a process- (or result-object or result-state) selecting predicate. In other words, each of these three operators returns a functor which takes a more specific argument; and so, an abstraction using another operator which expects the same sort of argument as χ, θ, or ω won't be able to combine with any of the results yielded by these three abstractions.

The meronymic abstraction ${}_{\preceq}\mathcal{F}_{\mu}^{\Rightarrow}$ is endocentric, since the operator μ (defined in (11)) yields an individual which is more general than its input individual and so can combine with a wider variety of functors (e.g., both process and result-object selecting functors) than the input individual. Further, it is a granularity abstraction under the indistinguishability relation $\sim_1$. ${}_{\preceq}F_{\mu}^{\Rightarrow}$ is part of a family of endocentric meronymic granularity abstractions which apply to nominalizations, including ${}_{\preceq}\mathcal{F}_{\vartheta}^{\Rightarrow}$ and ${}_{\preceq}\mathcal{F}_{\xi}^{\Rightarrow}$.

At this point, one may ask what has been gained by characterizing the type-shifting operators above as abstractions. We first address underspecification. It is useful to characterize underspecification over individuals as an abstraction instead of merely a type shift, because granularity seems clearly involved - in particular, the concept of **indistinguishability** (e.g., Definition 11), which is absent from type shifting, obviously applies.

The case for characterizing polysemy in terms of granularity is not very striking, but nevertheless there are intuitively appealing grounds for doing so. Let's say the grain size shifts, so that the discourse shifts from talking about an event to talking about a part of the event (causing, in our view, the meaning of polysemous nominalizations describing the event to become more fine-grained). Type shifts in themselves have no representation of the current "level of abstraction" of the discourse (and consequently of the reasoning processes), whereas the granularity characterization offers the concept of **grain size**. Tracking the grain size, in turn, can help constrain reasoning processes, as revealed in investigations of research on "task-oriented" dialog (e.g., [Grosz 77]).

The final reason for viewing these operations as abstractions is methodological. In thinking of them as abstractions and formalizing them, we are forced to specify a meta-theory of these operations, which can provide for an improved modularity in our reasoning systems. Such a meta-theory tells us how expressive the language for representing abstractions needs to be, provides a classification of abstraction, specifies what particular form abstractions can take, what their truth-conditional properties are, and what their effect on compositional structure is. Except for the first question, the others have been directly addressed in this paper.

6 RELATED WORK

This formalization of abstractions originates in the work of [Giunchiglia and Walsh 92], who consider abstractions to be mappings between formal systems, which consist of a language L, inference rules D, and axioms W. The abstractions defined here correspond roughly to what they call "L/W invariant" abstractions, where the axioms are not distinguished from the well-formed formulas of the language. There are several major points of difference, however. First of all, the discusssion in [Giunchiglia and Walsh 92] includes a syntactic classification of abstractions based on the structure of terms in the logical form language. For example, they distinguish Predicate Abstractions, which map predicate names, Domain Abstractions which map constants and function symbols, and Propositional Abstractions, where the number of arguments of a predicate can be decreased or increased. Such a syntactic classification is not useful for natural language logical forms unless the logical forms are canonicalized in some form, e.g., in the form of a Davidsonian representation where event predicate symbols are monadic. Second, our abstractions are of special interest to phenomena like natural language, where composition plays a critical role. Finally, we classify the set of abstractions in terms of particular types of inferential paths, including those involving $\subseteq$ and $\preceq$, making such abstractions very appropriate as inference mechanisms in ontologies.

A formalization of granularity was proposed by [Hobbs 85]. As we have pointed out earlier, following Giunchiglia and Walsh, Hobbs' notion of granularity shifts corresponds to a special class of granularity abstractions, some of which have been exemplified in this paper. There are also several other accounts of granularity shifts. [Euzenat 95] discusses granularity operators, which are mappings on relations in a ontology. In our case, abstractions have been defined as mappings among logical forms, with different kinds of abstractions being characterized in terms of their denotations. His account exploits complex relations which are disjunctions of other relations, which in our framework would correspond to **individual correlates** of the sets corresponding to the complex relations. Finally, in contrast to Euzenat's approach, which postulates a number of fundamental algebraic properties of granularity operators, our approach is based on having abstractions specify explicitly their truth-preserving properties.

Finally, there are references to the idea of granularity in other work. For example, [Pianesi and Varzi 96] discuss degrees of temporal granularity in event structure. The abstraction which shifts the perspective of *meeting* from (say) time intervals to specific instants, as illustrated in (4a), is characterized in their work in terms of a "minimal divisor" on structures corresponding to sets of events, where temporal differences within the divisor are neglected. It may be characterized in our approach in terms of a **sortal abstraction**. [Asher and Vieu 95] discuss perspective shifts in spatial domains; the shift of perspective in (4b) of "a pencil point" from a point to a surface could be characterized in terms of abstractions, but it first requires some representation of mereotopology (cf. [Asher and Vieu 95]), which we will examine in future work.

7 PRACTICAL APPLICATIONS

The motivation for this work is not entirely theoretical. In binding a NLP system to an application, the meanings of natural language expressions have to be mapped in a variety of different ways. A particular module may choose to collapse certain distinctions or to fill in and elaborate more detail for a particular application program. Providing a theory characterizing these mappings can remove a degree of adhocness and task-dependence in the design of the semantics/pragmatics interface in a wide variety of different tasks involving NLP, e.g., the business of constructing database queries, filling templates, generating a translation into a different natural language, and translating commands into directives to a simulation [Moore et al 96]).

The goal of developing robust ontologies capable of being used for inference in a variety of applications remains a challenge, since the concepts which need to be represented in different domains vary greatly, and the merging of disparate ontologies can be a formidable task. Nevertheless, several trends suggest that we are making progress towards this goal. First, standards have been evolving for knowledge exchange (e.g., [Gruber 93]) among different ontologies, as well as layered architectures for developing ontologies (e.g., [HPKB]). Second, there has also been progress on standardization of ontologies [ANSI]. Third, progress has been made in terms of a meta-theory of primitives used in knowledge representation systems [Guarino 94a], [Guarino 94b]. Fourth, there have emerged several large-scale ontologies [Mikrokosmos], [CYC], where a significant degree of reuse across applications is the norm rather than the exception. Fifth, relatively fine-grained thesauri such as [WordNet] (see also [Euro-WordNet]) continue to be used in different applications, although their use is not entirely problem-free. Finally, interesting techniques have been developed (e.g., [Knight and Luk 94]) for semi-automatic merging of diverse ontologies. All this suggests that some day in the forseeable future we will have rich large-scale semantic resources available, that can be reused across applications.

When used for lexical semantics, such a resource has the potential to give rise to massive ambiguity, unless we can develop generative theories which allow for dynamic determination of the set of relevant meanings by constraining our inferential processes to certain subontologies. Abstractions, defined as they are as ontological operations, provide a generative mechanism for deriving new granularity-shift-related meanings

This approach to granularity may be particularly useful in tying more fundamental concepts in an ontology to more domain-specific ones. For example, a natural language front-end to a battle simulation system (e.g., [Moore et al 96]) may map words like *attack* or *approach* to complex sequences of actions in the target simulation system (e.g., *Attack Checkpoint Charlie at 1500*). To reuse lexical information across multiple applications, it may be desirable to integrate a general English lexical semantic ontology (where *attack* may have several meanings, e.g., related to military or medical situations, etc.) with an ontology for the application (where *attack* maps to a particular battle simulation action). If the attack is to begin at a particular time (e.g., *Platoon 6, launch an attack at 0900 hours*), the event structure of the general meaning (i.e., that attacks are processes) is relevant. A simulation may treat attacks as instantaneous events, akin to achievements, and military units like Platoon 6 as primitive (point) objects, simplifying the specification of an attack, or it may be more fine-grained, treating attacks as involving a non-infinitesimal time interval, and units as having component objects, including soldiers, vehicles, etc. (which may in turn be realized as geometric shapes on a map). These mappings can be characterized as abstractions, which may be reused in different ways for an NL system which must talk to several different simulations (a not uncommon situation).

There are also other cases where this approach may bear particular fruit. For example, a particular (transfer-based) machine translation system (e.g., [Nagao 87]) may represent just enough word meaning in the source language to disambiguate its possible translations in a target language; a more interlingual approach focused on problems of translation divergences at the semantic level [Dorr 93a], [Dorr 93b], [Dorr et al 94] may use fairly abstract representations of meaning, such as Conceptual Structures (CS) [Jackendoff 83], [Jackendoff 90], [Jackendoff 91] (see also [Zwarts and Verkuyl 94]). In applying my approach, a particular semantic representation of *swim* used in a semantic-transfer oriented lexicon might be decomposed into more "abstract" semantic representation used in more "interlingual" lexicons. We may define (as in [Mani 97b]) an endocentric abstraction which maps swimming into a particular kind of going event, which in turn allows certain path and manner incorporation translation divergences [Talmy 85], [Barnett et al. 94] (e.g., *swim across the river* ~ (French) *traverser la riviere a la nage*) to be resolved. Thus, the same semantic lexicon could be used for multiple theoretical approaches.

8 CONCLUSIONS

This paper has illustrated how a theory of reasoning processes based on the notion of granularity can be used as a foundation to analyze certain problems in natural language semantics. Efficiencies in language use, reflected in phenomena of polysemy and underspecification, are viewed as mirroring efficiencies in underlying reasoning processes. The overall idea is that we may fold certain distinctions, or split meanings more finely, as the grain size changes, in order to carry out the inferences needed to communicate. As such, the paper shows one way in which further synergy between the fields of knowledge representation and natural language semantics can be achieved. The synthesis of ideas presented here is intended to stimulate further discussion, and is fairly open-ended in scope. In the future, we expect to continue to develop the theory, and applying it to problems of lexical semantics in the domain of spatial relations, in particular, exploiting current work in mereotopological representations.

Acknowledgements

I am grateful to Leo Obrst and two anonymous referees for helpful comments on earlier drafts of this paper.

References

[Allen 81] James Allen, "What's Necessary to Hide?: Modeling Action Verbs", Proceedings of the 19th ACL, Stanford, June 29-July 1, 1981.

[Allen 83] James Allen, "Maintaining Knowledge About Temporal Intervals", Communications of the ACM, Volume 26, Number 11, 1983.

[Allen 84] James Allen, "Towards a General Theory of Action and Time", Artificial Intelligence 23 (1984), 123-154.

[ANSI] ANSI Ad Hoc Group on Ontology Standards, see http://WWW-KSL.Stanford.edu /onto-std/index.html.

[Asher and Vieu 95] N. Asher and L. Vieu, "Towards a Geometry of Common Sense: A Semantics and Complete Axiomatization of Mereotopology", Proceedings of IJCAI-95, Montreal, 1995, pp. 846-852.

[Bach 86] Emmon Bach, "The Algebra of Events", Linguistics and Philosophy 9, 5-16.

[Barnett et al. 94] J. Barnett, I. Mani, and E. Rich, "Reversible Machine Translation", in T. Strzalkowski, ed., Reversible Grammar in Natural Language Processing, pp. 321-364, Kluwer Academic Publishers, 1994.

[Briscoe and Copestake 91] E. H. Briscoe and A. Copestake, "Sense Extensions as Lexical Rules", in D. Fass, E. Hinkelman and J. Martin, eds., Proceedings of the IJCAI Workshop on Computational Approaches to Non-Literatal Language, Sydney, Australia, pp. 12-20.

[Chierchia 82] Gennaro Chierchia, "Nominalization and Montague Grammar: A Semantics Without Types for Natural Languages", Linguistics and Philosophy 5 (1982), 303-354.

[Chierchia 84] Gennaro Chierchia, Topics in the Syntax and Semantics of Infinitives and Gerunds, Ph. Dissertation, University of Massachusetts, 1984.

[Church 41] Alonzo Church, "The Calculi of Lambda-Conversion", Annals of Mathematical Studies, Vol. 6, Princeton University Press, 1941.

[Davidson 67] Donald Davidson, "The Logical Form of Action Sentences". In N. Rescher, ed. "The Logic of Decision and Action", University of Pittsburgh Press, Pittsburgh, 1967.

[Davidson 69] Donald Davidson, "The Individuation of Events". In N. Rescher, et al, eds. Essays in Honor of Carl G. Hempel, D. Reidel, Dordrecht, 1969.

[Dorr 93a] Bonnie Dorr, "The Use of Lexical Semantics in Interlingual Machine Translation", Machine Translation 7, 135-193, 1992/3.

[Dorr 93b] Bonnie Dorr, "Interlingual Machine Translation: a parameterized approach", Artificial Intelligence 63 (1993) 429-492.

[Dorr and Voss 93] Bonnie Dorr and Clare Voss, "Machine Translation of Spatial Expressions: Defining the Relation between an Interlingua and a Knowledge Representation System", in Proceeedings of AAAI-93, Washington, DC, 1993.

[Dorr et al 94] Bonnie Dorr, Clare R. Voss, Eric Peterson, and Mike Kiker, "Concept- Based Lexical Selection", Proceedings of the AAAI Fall Symposium on MT Lexicons, 1994.

[Dowty 79] David Dowty, "Word Meaning and Montague Grammar", D. Reidel, Boston, 1979.

[Euro-WordNet] Report on Euro WordNet, see http://WWW- KSL.Stanford.EDU/onto-std/eurowordnet.pdf

[Euzenat 95] Jerome Euzenat, An Algebraic Approach to Granularity in Qualitative Time and Space Representation, Proceedings of IJCAI-95.

[Giunchiglia and Walsh 90] Fausto Giunchiglia and Toby Walsh, "Abstraction in AI", Proceedings of the Conference for the Society for the Study of Artificial Intelligence and Simulation of Behavior (AISB-90), 22-26.

[Giunchiglia and Walsh 92] Fausto Giunchiglia and Toby Walsh, "A Theory of Abstraction", Artificial Intelligence, 57, 2-3, 323-390.

[Grosz 77] B. Grosz, "The Representation and Use of Focus in Dialog Understanding", Stanford Research International Technical Note 151, Menlo Park, CA, July 1977.

[Gruber 93] T. R. Gruber, A Translation Approach to Portable Ontology Specifications, Knowledge Acquisition, 5(2):199-220, 1993. Available at ftp://ksl.stanford.edu/pub/knowledge-sharing/papers/.

[Guarino 94a] Guarino N. The Ontological Level. Invited paper Presented at IV Wittgenstein Symposium, Kirchberg, Austria, 1993. In R. Casati, B. Smith and G. White (eds.), Philosophy and the Cognitive Sciences, Vienna, Hlder-Pichler-Tempsky 1994.

[Guarino 94b] Guarino N., Carrara M., Giaretta P. An Ontology of Meta-Level Categories. KR'94, Bonn, 1994.

[Hobbs 85] Jerry Hobbs, "Granularity", Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-85), 1985.

[HPKB] DARPA High-Performance Knowledge Base Program, http://www.teknowledge.com /HPKB/about/HPKBpipGoals.html.

[Hwang and Schubert 93] C. H. Hwang and L. K. Schubert, Episodic Logic: A Situational Logic for Natural Language Processing, in P. Aczel, D. Israel, Y. Katagirm and S. Peters, eds., Situation Theory and its Applications, Vol. 3, CSLI Lecture Notes No. 37, Center for the Study of Language and Information, Stanford University, 1993.

[Jackendoff 83] Ray Jackendoff, "Semantics and Cognition", MIT Press, Cambridge, MA, 1983.

[Jackendoff 90] Ray Jackendoff, "Semantic Structures", MIT Press, Cambridge, MA, 1990.

[Jackendoff 91] Ray Jackendoff, "Parts and Boundaries", Cognition, 41, (1991) 9-45

[Kamp 84] Hans Kamp, "A Theory of Truth and Semantic Representation", in J.A. G. Groenendijk, T.M. V. Janssen and M.J.B. Stokhof, eds., Formal Methods in the Study of Language, Dordrecht, Foris, 1984.

[Knight and Luk 94] Kevin Knight and Steve K. Luk, "Building a Large-Scale Knowledge Base for Machine Translation", Proceeedings of AAAI-94, Seattle, 1994.

[Krifka 87] Manfred Krifka, "Nominal Reference and Temporal Constitution: Towards a Semantics of Quantity", in J.A. G. Groenendijk, M.J.B. Stokhof, and F. Veltman, eds., Proceedings of the 6th Amsterdam Colloquium, 1987, 153-173.

[Landman 89] F. Landman, "Groups", Linguistics and Philosophy, 12, I:559-605, II: 723-744.

[CYC] Lenat, D. and Guha, R.V. Building Large Knowledge-based Systems: Representation and Inference in the CYC Project. Addison Wesley, 1990.

[Link 83] G. Link, The Logical Analysis of Plurals and Mass Terms: A Lattice-Theoretic Approach, in R. Bauerle, C. Schwarze, and A. von Stechow, eds., Meaning, Use, and Interpretation of Language, Mouton, Berlin, 1983.

[Mani 97a] Inderjeet Mani, "Underspecification in the Semantics of Nominalizations", Lexicalische Ambiguitaet und semantische Unterspezifikation, 1997 Annual Conference of the German Linguistic Society, Dusseldorf, February 26-28, 1997.

[Mani 97b] Inderjeet Mani, "The Semantics of Nominalizations: A Study in Decompositional Semantics", Ph.D. Dissertation, Georgetown University, Washington, D.C., February 1997.

[Mikrokosmos] Stephen Beale, Sergei Nirenburg and Kavi Mahesh, Semantic Analysis in The Mikrokosmos Machine Translation Project, Proc. Symposium on NLP, Kaset Sart University, Bangkok, Thailand, 1995.

[WordNet] Miller, G., "WordNet: A Lexical Database for English", Communications of the ACM, 38, 11, pp. 39-41, 1995.

[Moore et al 96] Robert Moore, John Dowding, Harry Bratt, J. Mark Gawron, Yonael Gorfu, and Adam Cheyer, "CommandTalk: A Spoken-Language Interface for Battlefield Simulations", Proceedings of the 5th Conference on Applied Natural Language Processing, 31 March -3 April 1997.

[Nagao 87] Makoto Nagao, "Role of Structural Transformation in a Machine Translation System". In

Sergei Nirenburg, ed., Machine Translation, Theoretical and Methodological Issues, Cambridge University Press, 1987.

[Parsons 90] Terence Parsons, "Events in the Semantics of English", MIT Press, Cambridge, 1990.

[Pianesi and Varzi 96] Fabio Pianesi and Achille C. Varzi, Refining Temporal Reference in Event Structures, Notre Dame Journal of Formal Logic, 37, 1, Winter 1996.

[Pustejovsky 95] James Pustejovsky, The Generative Lexicon, MIT Press, Cambridge, MA, 1995.

[Reyle 93] Uwe Reyle, "Dealing with Ambiguities by Underspecification: Construction, Representation, and Deduction", Journal of Semantics, 10, 123-179, 1993.

[Talmy 85] Leonard Talmy, "Lexicalization Patterns", in T. Shopen, ed., Language Typology and Syntactic Description, Volume III, Cambridge University Press, 1985.

[Vendler 67] Zeno Vendler, Linguistics in Philosophy, Cornell University Press, Ithaca, 1967.

[Zwarts and Verkuyl 94] "An Algebra of Conceptual Structure: An Investigation into Jackendoff's Conceptual Semantics". Linguistics and Philosophy 17: 1-28, 1994.

[Zucchi 93] Alessandro Zucchi, The Language of Propositions and Events, Kluwer, 1993.

Belief Revision and Contextual Reasoning

The PMA and Relativizing Change for Action Update

Patrick Doherty
Dept. of Computer Science
Linköping University
S-58183 Linköping, Sweden
patdo@ida.liu.se

Witold Łukaszewicz
Ewa Madalińska-Bugaj
Institute of Informatics
Warsaw University
Warsaw, Poland

Abstract

Using intuitions from the temporal reasoning community, we provide a generalization of the PMA, called the modified PMA (MPMA), which permits the representation of disjunctive updates and the use of integrity constraints interpreted as causal constraints. In addition, we provide a number of syntactic characterizations of the MPMA, one of which is constructed by mapping an MPMA update of a knowledge base into a temporal narrative in a simple temporal logic (STL). The resulting representation theorem provides a basis for computing entailments of the MPMA and could serve as a basis for further generalization of the belief update approach for reasoning about action and change.

1 Introduction

Recently, much effort has been invested in applying the belief revision/update (BR/U) paradigm to the domain of reasoning about action and change. In a similar vein, much effort has been invested in applying temporal logics (TL) to reasoning about action and change. Unfortunately, there have been few attempts at trying to bridge the gap between the two paradigms by analyzing one approach in terms of the evaluation methodology of the other, or by applying solutions generated in one paradigm to similar open problems in the other paradigm (although, see [10, 20, 3, 24, 22]). We believe that there is much to be gained by a cross-fertilization of the two paradigms and in this paper, we will try to do just that.

We begin by noting that much of the recent research in the temporal reasoning paradigm has focused on two issues: 1) proper representation of non-deterministic actions [18, 21, 15], and 2) proper representation of the indirect effects of actions, where the use of causal or fluent dependency constraint approaches has received a great deal of attention [17, 19, 13, 23]. On the other hand, the BR/U approach applied to reasoning about action and change, does not appear to have reached any satisfactory consensus on the proper approach to disjunctive update and is continually plagued with an inability to represent casual constraints in terms of standard integrity constraints.

Beginning with the PMA, we will apply techniques from the TL paradigm resulting in a generalization of the PMA, called the modified PMA (MPMA), which permits the representation of disjunctive update together with integrity constraints interpreted as causal constraints. In addition, we provide a number of syntactic characterizations of MPMA with integrity constraints and show that a very simple temporal logic, STL, which is a fragment of a 1st-order temporal logic described in [4, 13] with origins from the Features and Fluents framework [21], can be used to compute entailments of the MPMA with integrity constraints. The representation theorem we provide opens up the possibility of additional generalizations of the BR/U paradigm applied to reasoning about action and change.

Another interesting result of this work is the discovery that several existing approaches to reasoning about action and change [15, 23, 13], suffer from a form of syntactic sensitivity which we call the *redundant atom anomaly*, where if one is not careful about the syntactic form of consequents to action or causal rules, certain arguably unintuitive direct and indirect effects of action invocation may result. In the paper, we point out the problem using MPMA, and provide a remedy which can also be applied to the temporal logics which suffer from similar problems.

The generalization to the PMA can succinctly be described as the *relativization* or weakening of the minimal change policy to a subset of atoms in the language rather than to all atoms. The technique is similar to the use of varied predicates in circumscription or the use of an *occlusion*, *release*, or *noninert* predicate in the temporal logic paradigm. The proposed solution to the use of integrity constraints is to view them as causal constraints representing fluent dependency information as described in Gustafsson & Doherty [13], where the technique was used to deal with the ramification problem.

The paper is structured as follows: In Section 2, we introduce the PMA and discuss some well known criticisms associated with its use as a formalism for reasoning about action and change. In Section 3, we modify the PMA, by relativizing or weakening the minimal change policy built into the definition of distance between interpretations. We discuss the redundant atom anomaly and provide a means of automatically preprocessing updates to remove redundancy and choose the atoms to be excluded from the minimal change policy. A syntactic characterization of MPMA in terms of *eliminants* is considered along with an analysis of the MPMA in terms of the Katsuno and Mendelzon [14] postulates. In Section 4, we introduce the simple temporal logic, STL, a fragment of a highly expressive logic used to reason about temporal narratives. In Section 5, we reformulate the MPMA in STL and provide a representation theorem. A decision procedure for MPMA in terms of STL is then provided. In Section 6, we extend the MPMA with integrity constraints represented as causal rules or dependency constraints.In Section 7, we provide a comparison with related work and in Section 8, conclude with a discussion.

Due to page limitations, proofs are excluded, but may be found in Doherty *et al.* [6], an extended version of this paper which also includes an additional syntactic characterization of the MPMA in terms of a Dijkstra semantics based approach together with a corresponding decision procedure, and additional comparisons with related work.

2 The Classical PMA

We present the classical PMA semantics, originally developed by Winslett [25, 26, 27].

2.1 The Language $\mathcal{L}_{pma}$

We start with a language $\mathcal{L}_{pma}$ of classical propositional logic based on a finite fixed set $ATM = \{p, q, r, \ldots\}$ of atoms. Formulas are built in the usual way using the connectives $\wedge, \vee, \neg, \rightarrow, \leftrightarrow, \top$ (truth) and $\perp$ (falsity). If α and β are formulas and p is an atom, then we write $\alpha[p \leftarrow \beta]$ to denote the formula which is obtained from α by replacing all occurrences of p by β. A *literal* is an atom or its negation.

2.2 Semantics

Interpretations are identified with maximal consistent sets of literals. For any formula α, we write $|\alpha|$ to denote the set of all *models* of α, *i.e.* interpretations satisfying α. A formula α is said to *correspond* to an interpretation u iff $|\alpha| = u$. To construct such a formula, it suffices to take the conjunction of all literals occurring in u. Similarly, a formula α is said to correspond to a finite set of interpretations $\{u_1, \ldots, u_n\}$ if $|\alpha| = \{u_1, \ldots, u_n\}$. To obtain such a formula, it suffices to take the disjuncion $\beta_1 \vee \cdots \vee \beta_n$, where, for each $1 \leq i \leq n$, β_i is the formula corresponding to u_i. For instance, the formula corresponding to the set $\{\{p, \neg q\}, \{p, q\}\}$ is $(p \wedge \neg q) \vee (p \wedge q)$ which is equivalent to p.

Definition 1 Let w, v be two interpretations. The *distance* between w and v, written $DIST(w, v)$, is a set of atoms that have different truth-values in w and u. ■

For instance, the distance between $w = \{p, q, r, \neg s\}$ and $v = \{q, s, \neg p, \neg r\}$ is $\{p, r, s\}$.

Definition 2 The *update of an interpretation w by a set of interpretations V*, written $w \star V$, is the set of those elements of V that are closest to w, *i.e.* whose distance to w is minimal. More formally:

$$w \star V = \{v \in V : \text{there is no } v' \in V \text{ such that } DIST(w, v') \subset DIST(w, v)\}. \blacksquare$$

For instance,
if $w = \{\neg p, \neg q\}$ and $V = \{\{p, \neg q\}, \{q, \neg p\}, \{p, q\}\}$, then $w \star V = \{\{p, \neg q\}, \{q, \neg p\}\}$.

Definition 3 The *update of a set of interpretations U by a set of interpretations V*, written $U \star V$, is given by

$$U \star V = \bigcup_{w \in U} w \star V. \blacksquare$$

In the classical PMA, the update $KB \star \alpha$ of a knowledge base KB by a formula α is identified with the formula corresponding to the set of interpretations $|KB| \star |\alpha|$.

2.3 Examples

Example 1 Let $ATM = \{p, q\}$, $KB = p$ and $\alpha = p \vee q$. Then $|KB| \star |\alpha| =$

$$\{p, \neg q\} \star \{\{p, \neg q\}, \{q, \neg p\}, \{p, q\}\} \cup \{p, q\} \star \{\{p, \neg q\}, \{q, \neg p\}, \{p, q\}\} = \{p, \neg q\} \cup \{p, q\}.$$

Thus $KB \star \alpha$ is the formula $(p \wedge \neg q) \vee (p \wedge q)$ which is equivalent to p.

Example 2 Let $ATM = \{p, q\}$, $KB = \neg p \wedge \neg q$ and $\alpha = p \vee q$. Then $|KB| \star |\alpha| =$

$$\{\neg p, \neg q\} \star \{\{p, \neg q\}, \{q, \neg p\}, \{p, q\}\} = \{\{p, \neg q\}, \{q, \neg p\}\}.$$

Thus $KB \star \alpha$ is the formula $(p \wedge \neg q) \vee (\neg p \wedge q)$.

3 The Modified PMA

A number of researchers ([1], [11]) have observed that the classical PMA may lead to unintuitive results, particularly when disjunctive update is involved. To see this, reconsider Example 1. If all we know about the world is p and the effect of an action performed is $p \vee q$, then the description of the new state of the world should be $p \vee q$ rather than p. If p was accepted as the description of the new state, then this would imply that our knowledge about the current state makes an unpredictable action predictable. Example 2 also illustrates an undesirable phenomenon: updating a knowledge base by the inclusive *or* results in the exclusive *or*. What these two examples have in common is the fact that if an action effect follows from the database, the action will have no effect. The culprit in both cases is the fact that the PMA minimal change policy should be relaxed somewhat, but in a principled manner. This is the insight gained from the TL community and incorporated in current solutions to similar problems.

In order to incorporate a relaxed minimal change policy for the PMA, we will change it in two ways. Firstly, the distance between two interpretations will always be relativized to a chosen set of atoms. Intuitively, this set represents atoms that are allowed to vary their values during the action execution. Secondly, the update of an interpretation w by a set of interpretations U will be defined as those elements of U whose distance to w is the empty set. We call this new form of the PMA the *modified* PMA and we denote it by MPMA. The details follow.[1]

Definition 4 Let w and v be two interpretations and suppose that P is a set of atoms. The *distance* between w and v wrt P is $DIST(w, v) - P$. ■

For instance, the distance between $\{p, q, r, \neg s\}$ and $\{q, s, \neg p, \neg r\}$ wrt $\{p\}$ is $\{r, s\}$.

Definition 5 The *update of an interpretation w by a set of interpretations V* wrt a set of atoms P, written $w \star^P V$, is the set of those elements of V whose distance to w wrt P is $\emptyset$. ■

Definition 6 The *update of a set of interpretations U by a set of interpretations V* wrt a set of atoms P, written $U \star^P V$, is given by

$$U \star^P V = \bigcup_{w \in U} w \star^P V. \blacksquare$$

In the modified PMA, the update $KB \star^P \alpha$ of a knowledge base KB by a formula α wrt a set of atoms P is identified with the formula corresponding to the set of interpretations $|KB| \star^P |\alpha|$.

Example 1 (continued) To properly deal with this example, the atoms p and q should be allowed to vary. $|KB| \star^{\{p,q\}} |\alpha| =$

$$\{p \neg q\} \star^{\{p,q\}} \{\{p, \neg q\}, \{q, \neg p\}, \{p, q\}\} \cup \{p, q\} \star^{\{p,q\}} \{\{p, \neg q\}, \{q, \neg p\}, \{p, q\}\} =$$
$$\{\{p, \neg q\}, \{q, \neg p\}, \{p, q\}\} \cup \{\{p, \neg q\}, \{q, \neg p\}, \{p, q\}\} = \{\{p, \neg q\}, \{q, \neg p\}, \{p, q\}\}.$$

Thus $KB \star^{\{p,q\}} \alpha$ is the formula $(p \wedge \neg q) \vee (q \wedge \neg p) \vee (p \wedge q)$ which is equivalent to $p \vee q$. ■

Example 2 (continued) As before, the variable atoms should be p and q. We leave it to the reader to check that $KB \star^{\{p,q\}} \alpha \equiv p \vee q$. ■

It should be noted that the MPMA need not preserve consistency. For instance, if $ATM = \{q\}$, $KB = \neg q$, $\alpha = q$ and $P = \emptyset$, then $KB \star^P \alpha \equiv \bot$. However, as we shall see in the next section, the consistency preservation is guaranteed if variable atoms are properly chosen.

3.1 Determining Variable Atoms

In this section we discuss the fundamental question: how can one provide a mechanism to determine vari-

[1]The motivation behind MPMA will become clear when we provide its syntactic characterization in Section 3.2. A policy similar to MPMA is mentioned by Winslett, but rejected as not representing minimal change. MPMA also reflects similar minimization policies used in the TL paradigm as we will show.

able atoms which can be applied automatically and still guarantee intuitively reasonable conclusions?

Consider a knowledge base KB and an update formula α. Since α represents effects of a performed action, all atoms occurring in α are potential candidates for variable atoms. However, one should be cautious. The reason is that some atoms from α may be redundant. For example, if α is $p \vee (p \wedge q)$, then $\alpha \equiv p$ and so q should not be considered as a variable atom.

Example 3 Suppose that $ATM = \{p, q\}$, $KB = \neg p \wedge q$ and $\alpha = p$. Taking p as the only variable atom, we get $KB \star^{\{p\}} \alpha \equiv p \wedge q$.

Suppose now that we introduce a redundant atom to α, replacing α by its equivalent $\alpha' = (p \wedge q) \vee (p \wedge \neg q)$. Taking p and q as variables, we obtain $KB \star^{\{p,q\}} \alpha' \equiv p$. This result is clearly undesirable. ■

This is an example of what we call the *redundant atom anomaly*. It appears to be a flaw that several formalisms [15, 17, 13], based on the use of an *Occlusion* or *Release* predicate, succumb to. *Occlusion* and *Release* have the role of distinguishing variable fluents from non-variable fluents. In fact, it can be shown that other causal formalisms such as Thielscher [23] suffer from the same flaw when syntactic rules are not preprocessed to remove redundant atoms. Semantic approaches toward modeling action effect axioms such as Sandewall [21], where a *full trajectory normal form* (FTNF) is generated from a model-theoretic description, do not suffer from this problem because there are no redundant atoms in the normal form. On the other hand, the claim that the FTNF can be replaced with a logically equivalent formula does not appear to be correct for those logics dependent on the use of an *Occlude* predicate, where the choice of occluded fluents is dependent on syntax.

Definition 7 Let α be a formula. An atom p occurring in α is said to be *redundant* for α iff $\alpha[p \leftarrow \top] \equiv \alpha[p \leftarrow \bot]$.[2] ■

As follows from the above definition, an atom is redundant for a formula iff the logical value of the formula does not depend on the logical value of the atom.

Our choice of variable atoms is to identify them with the set of non-redundant atoms for the update formula. This poses the question of how these atoms are to be selected. Definition 7 is rather impractical from the computational point of view. Below, we present a generally more efficient method based on the notion of the *Blake canonical form* of a formula. Our discussion follows Brown [2].

We start with preliminary terminology.

A *term* is either $\top$, $\bot$, or a conjunction of literals in which no atom appears more than once. A formula is said to be in *disjunctive normal form* (*DNF*, for short) if it is a disjunction of different terms.[3] It is well-known that each formula can be constructively transformed into its equivalent in *DNF*. We say that a term t_1 *absorbs* a term t_2 if either t_1 is $\top$, or t_2 is $\bot$, or t_1 is a subterm of t_2. For instance, the term p absorbs the term $p \wedge q$. Let α be a formula in *DNF*. We write $ABS(\alpha)$ to denote the formula obtained from α by deleting all absorbed terms. Clearly, α and $ABS(\alpha)$ are equivalent.

Two terms are said to have an *opposition* if one of them contains an atom p and the other the atom $\neg p$. For instance, the terms $\neg q \wedge r$ and $q \wedge s$ have a single opposition, in the atom q.

Suppose that two terms, t_1 and t_2, have exactly one opposition. Then the *consensus* of t_1 and t_2, written $c(t_1, t_2)$, is the term obtained from the conjunction $t_1 \wedge t_2$ by deleting the opposed atoms as well as any repeated atoms. For example, $c(\neg q \wedge p, q \wedge r)$ is $p \wedge r$.

Let α be a formula. The *Blake canonical form of* α, written $BCF(\alpha)$, is the formula obtained from α by the following construction.

(1) Replace α by its disjunctive normal form. Denote the resulting formula by β.

(2) Repeat as long as possible:
If β contains a pair t_1 and t_2 of terms whose consensus exists and no term of β is a subformula of $c(t_1, t_2)$, then $\beta := \beta \vee c(t_1, t_2)$.

(3) Take $ABS(\beta)$. This is $BCF(\alpha)$.

The following results can be found in Brown [2].

Theorem 1
(1) Formulas α and $BCF(\alpha)$ are equivalent.
(2) All atoms occurring in $BCF(\alpha)$ are non-redundant for α. ■

[2]Recall that $\alpha[p \leftarrow \top]$ (resp. $\alpha[p \leftarrow \bot]$) is the formula obtained from α by replacing all occurrences of p by $\top$ (resp. $\bot$).

[3]In the logical literature *DNF* is often defined as a disjunction of terms where a term is understood as either $\top$, or $\bot$ or any conjunction of literals. Note, however, that we can always restrict ourselves to terms in which no atom appears more than once: each repeated occurence of an atom p can be removed from a term, whereas any term including p and $\neg p$ can be replaced by $\bot$.

Example 4 Let α be $(\neg p \wedge q) \vee (p \wedge r \wedge \neg s) \vee (p \wedge \neg s \wedge \neg q)$. Since α is in disjunctive normal form, $\beta = \alpha$. After performing step (2), the result is,

$$(\neg p \wedge q) \vee (p \wedge r \wedge \neg s) \vee (p \wedge \neg s \wedge \neg q) \vee (r \wedge q \wedge \neg s). \quad (1)$$

Since $ABS((1)) = (1)$, the formula (1) is the Blake canonical form of α. ■

As we noted in the previous section, the MPMA need not preserve consistency. However, if all non-redundant atoms of the update formula are considered as variable atoms, consistency preservation is assured. In the sequel, we write $atm(\alpha)$ to denote the set of all atoms occurring in a formula α.

Theorem 2 For any knowledge base $KB \not\equiv \perp$ and any formula $\alpha \not\equiv \perp$, $P = atm(BCF(\alpha))$ implies $KB *^P \alpha \not\equiv \perp$. ■

3.2 Syntactic Characterization of the MPMA

In this section, we present a syntactic characterization of the MPMA. We start with some terminology.

Let p be an atom and suppose that α is a formula. We write $\exists p.\alpha$ to denote the formula $\alpha[p \leftarrow \top] \vee \alpha[p \leftarrow \perp]$. If $P = \{p_1, \ldots, p_n\}$ is a set of atoms and α is a formula, then $\exists P.\alpha$ stands for $\exists p_1 \cdots \exists p_n.\alpha$.

A formula of the form $\exists P.\alpha$, where $P = \{p_1, \ldots, p_n\}$, is called an *eliminant of* $\{p_1, \ldots, p_n\}$ *in* α. Intuitively, such an eliminant can be viewed as a formula representing the same knowledge as α about all atoms from $ATM - P$ and providing no information about the atoms in P. The reader interested in a detailed theory of eliminants should consult Brown [2].

The following result, which easily follows from the definition of an eliminant, provides its semantical characterization.

Theorem 3 Let α be a formula and suppose that P is a set of atoms. Then $|\exists P.\alpha| =$

$\{u :$ exists $w \in |\alpha|$ such that $DIST(w, u) - P = \emptyset\}$. ■

We are now ready to provide a syntactic characterization of the MPMA.

Theorem 4 Let KB, α and $P = atm(BCF(\alpha))$ be a knowledge base, a formula and a set of atoms, respectively. Then $KB \star^P \alpha \equiv \alpha \wedge \exists P.KB$. ■

Theorem 4 clearly shows how the MPMA works. First, we select the atoms that may vary their values when the action corresponding to the update formula α is performed. Next, we weaken the knowledge base KB by eliminating all those variable atoms. Finally, we strengthen the knowledge base $\exists P.KB$ by combining it with the update formula.

3.3 Computing the MPMA

In view of Theorem 4, all we need to compute the MPMA is the ability to compute eliminants. The following method can be found in Brown [2].

Let α be a formula and suppose that $P = \{p_1, \ldots, p_k\}$ is a set of atoms. The eliminant $\exists P.\alpha$ is the formula obtained by the following construction.

(1) Replace α by its disjunctive normal form $t_1 \vee \cdots \vee t_n$, where $t_1, \ldots, t_n$ are terms. Denote the resulting formula by β.

(2) From each t_i $(1 \leq i \leq n)$, remove all occurrences (both positive and negative) of $p_1, \ldots, p_k$. If during this process all literals have been removed from some term t_i, stop and return $\top$ as $\exists P.\alpha$. Otherwise, return the resulting formula as $\exists P.\alpha$.

Example 5 Let $\alpha = (p \rightarrow q) \wedge (\neg q \vee r)$. We compute $\exists q.\alpha$. Converting α into its disjunctive normal form, we get $\beta = (\neg p \wedge \neg q) \vee (\neg p \wedge r) \vee (q \wedge r)$. Eliminating the atom q, we finally obtain $\neg p \vee (\neg p \wedge r) \vee r$ which is equivalent to $\neg p \vee r$.

Example 6 This is a classical example from Winslett [25]. We have two atoms, b and m, standing for "a book is on the floor" and "a magazine is on the floor", respectively. $KB = (b \wedge \neg m) \vee (\neg b \wedge m)$. The update formula α is m. Since α is already in the Blake canonical form, the only variable atom is m. We first compute $\exists m.KB$. KB is already in disjunctive normal form. Eliminating all occurrences of m in KB, we get $b \vee \neg b$ which is equivalent to $\top$. Hence, $KB *^m \alpha$ is $m \wedge \top$ which is equivalent to m. ■

3.4 Properties of the MPMA

Katsuno & Mendelzon [14] proposed eight postualtes which, as they claimed, should be satisfied by each update operator. In this section, we analyse the MPMA in the context of these postulates.

Let KB be a knowledge base and suppose that α is an update formula. In what follows, we shall write $KB \star \alpha$ as an abbreviation for $KB \star^P \alpha$, where P is the set of all atoms occurring in the Blake canonical form of α.

Katzuno and Mendelzon set up the following postulates:

(1) $KB \star \alpha$ implies α.

(2) If KB implies α, then $KB \star \alpha$ is equivalent to KB.

(3) If both KB and α are satisfiable, then $KB \star \alpha$ is also satisfiable.

(4) If $KB_1 \equiv KB_2$ and $\alpha_1 \equiv \alpha_2$, then $KB_1 \star \alpha_1 \equiv KB_2 \star \alpha_2$.

(5) $(KB \star \alpha_1) \wedge \alpha_2$ implies $KB \star (\alpha_1 \wedge \alpha_2)$.

(6) If $KB \star \alpha_1$ implies α_2 and $KB \star \alpha_2$ implies α_1, then $KB \star \alpha_1 \equiv KB \star \alpha_2$.

(7) If KB is complete, *i.e.* has at most one model, then $(KB \star \alpha_1) \wedge (KB \star \alpha_2)$ implies $KB \star (\alpha_1 \vee \alpha_2)$.

(8) $(KB_1 \vee KB_2) \star \alpha \equiv (KB_1 \star \alpha) \vee (KB_2 \star \alpha)$.

We have already seen (Example 1 (continued)) that postulate (2) does not generally hold in MPMA. This is also the case as regards postulates (5), (6) and (7).

To see that (5) does not generally hold, take $ATM = \{m, b\}$, $KB = \neg m \wedge b$, $\alpha_1 = \neg b \vee m$ and $\alpha_2 = b \vee m$. $(KB \star \alpha_1) \wedge \alpha_2 = (\neg b \vee m) \wedge (b \vee m) \equiv m$. On the other hand, $KB \star (\alpha_1 \wedge \alpha_2) \equiv KB \star m \equiv b \wedge m$. Clearly, m does not imply $b \wedge m$. We claim that our result is intuitively plausible. Let us interpret b and m as "a book is on the table" and "a magazine is on the table", respectively. KB states that initially the magazine is not on the table and the book is on it. α_1 corresponds to the action "remove the book from the table or put the magazine on it (or perform both these subactions together). It is obvious that the updated knowledge base should be $\neg b \vee m$. If we take the conjunction of the new knowledge base and α_2, we will obtain m. On the other hand, $\alpha_1 \wedge \alpha_2$ is equivalent to m, which means that the action it corresponds to is "put the magazine on the table". Now, if we apply this action to our original knowledge base, the updated knowledge base should be $m \wedge b$.

It turns out that the following weaker form of postulate (5) holds for MPMA:

Theorem 5 If $atm(BCF(\alpha_1)) \subseteq atm(BCF(\alpha_1 \wedge \alpha_2))$, then $(KB \star \alpha_1) \wedge \alpha_2$ implies $KB \star (\alpha_1 \wedge \alpha_2)$. ■

Intuitively, Theorem 5 states that postulate (5) holds, provided that the action corresponding to $\alpha_1 \wedge \alpha_2$ may vary every atom that can be varied by the action corresponding to α_1.

To see that (6) need not hold, suppose that $ATM = \{b, m\}$, $KB = \neg b \wedge m$, $\alpha_1 = (b \vee m)$ and $\alpha_2 = \top$. $KB \star \alpha_1 \equiv b \vee m$ implies α_2. Similarly, $KB \star \alpha_2 \equiv \neg b \wedge m$ implies α_1. On the other hand, $KB \star \alpha_1$ and $KB \star \alpha_2$ are not equivalent. We leave it to the reader to check that the interpretation of the atoms b and m as "a book is on the table" and "a magazine is on the table", respectively, makes our result plausible.

The following weaker form of postulate (6) holds for the MPMA:

Theorem 6 Let $atm(BCF(\alpha_1)) = atm(BCF(\alpha_2))$. If $KB \star \alpha_1$ implies α_2 and $KB \star \alpha_2$ implies α_1, then $KB \star \alpha_1 \equiv KB \star \alpha_2$. ■

Intuitively, Theorem 6 states that postulate (6) holds, provided that the actions corresponding to α_1 and α_2 may vary exactly the same atoms.

To see that (7) need not to hold, suppose that $ATM = \{b, m, n\}$, $KB = m \wedge b \wedge n$, $\alpha_1 = (b \wedge m) \vee n$ and $\alpha_2 = (b \wedge \neg m) \vee n$. It is easily verified that $(KB \star \alpha_1) \wedge (KB \star \alpha_2) = \alpha_1 \wedge \alpha_2$ which is equivalent to n. On the other hand, $\alpha_1 \vee \alpha_2$ reduces to $b \vee n$, when the redundant atom m is eliminated. So, $KB \star (\alpha_1 \vee \alpha_2)$ is equivalent to $(b \vee n) \wedge m$ which is not a consequence of n. We leave it to the reader to check that our result is plausible if the atoms b, m and n are interpreted as "a book is on the table", "a magazine is on the table" and "a newspaper is on the table", respectively.

The following version of postulate (7) holds for the MPMA:

Theorem 7 If $atm(BCF(\alpha_1) \subseteq atm(BCF(\alpha_1 \vee \alpha_2))$ and $atm(BCF(\alpha_2) \subseteq atm(BCF(\alpha_1 \vee \alpha_2))$, then $(KB \star \alpha_1) \wedge (KB \star \alpha_2)$ implies $KB \star (\alpha_1 \vee \alpha_2)$.[4] ■

Intuitively, Theorem 7 says that postulate (7) holds, provided that the set of variable atoms of the action corresponding to α_1 and the set of variable atoms of the action corresponding to α_2 are both subsets of the set of variable atoms of the action corresponding to $\alpha_1 \vee \alpha_2$.

The remaining postulates of Katsuno & Mendelzon hold for the MPMA:

Theorem 8 The MPMA satisfies postulates (1), (3)-(4) and (8). ■

4 A Simple Temporal Logic

Since our primary concern in this section is to compare the knowledge-base update and temporal logic

[4]The assumption that KB is complete is not necessary here.

approaches to reasoning about action and change, we will only be concerned with action scenarios specified in terms of an initial state description S, a formula α describing an effect of an action executed in S and a resulting state description S'. This will correlate with an initial KB, an update formula α added to the KB, and the resulting KB. The above restrictions allow us to work with a very simple pseudo-temporal logic, referred to as STL. Actually, STL is the classical propositional logic which simulates a fragment of the 1st-order temporal logic in Doherty [4, 13], restricted to two time-points.

4.1 The Language $\mathcal{L}_{stl}$

Given the language $\mathcal{L}_{pma}$, based on a set of atoms ATM, we define the language $\mathcal{L}_{stl}$ as that of classical propositional logic over ATM_{stl}, where

$$\begin{aligned} ATM_{stl} = \{p^I : p \in ATM\} \cup \\ \{p^R : p \in ATM\} \cup \{O^p : p \in ATM\}. \end{aligned}$$

In other words, in the language $\mathcal{L}_{stl}$, each atom $p \in ATM$ is replaced by two copies p^I and p^R. In addition, for each atom $p \in ATM$, we introduce an auxiliary atom O^p. Intuitively, p^I and p^R represent the values of p in the initial state and in the resulting state, respectively. The atom O^p informally states that p is *occluded*, where occlusion plays the role of releasing an atom from keeping its value persistent from the initial state to the final state.

By an *I-formula* (resp. *R-formula*) we mean any formula constructable using the atoms of the form p^I (resp. p^R). For any formula $\alpha \in \mathcal{L}_{pma}$, we write α^I (resp. α^R) to denote the result of replacing each atom symbol p in α by p^I (resp. p^R).

The following straightforward result will be useful.

Proposition 1 For any $\alpha \in \mathcal{L}_{pma}$ and any $\beta \in \mathcal{L}_{pma}$, $\alpha \models \beta$ iff $\alpha^R \models \beta^R$, where $\models$ denotes the entailment relation of classical propositional logic. ■

Let α be an R-formula and suppose that $p_1^R, \ldots, p_n^R$ are all the atoms occurring in α. We write $Occlude(\alpha)$ as an abbreviation for $O^{p_1} \wedge \cdots \wedge O^{p_n}$.

4.2 Action Scenarios

Action scenarios, or narratives, are often used in the TL paradigm to represent sequences of action occurrences and observations together with a set of timing constraints which provide a partial or total order on the action occurrences. One then proposes an entailment policy, views the narrative as a theory and deduces facts about the narrative. Below, we provide a lightweight version of scenarios appropriate for our two state sequences.

Definition 1 (STL State Description) An *initial state description* is any I-formula. ■

Definition 2 (STL Action Description) An *action description* is the conjunction $\alpha \wedge Occlude(\alpha)$, where α is any R-formula. ■

Definition 3 (Action Scenario) An *action scenario* $\mathcal{A}$ is the conjunction $\alpha \wedge \beta$, where α is an initial state description and β is an action description. ■

4.3 The Minimization Policy

The minimization policy for STL is based on the PMON policy introduced by Sandewall [21] and investigated and extended in Doherty *et al.* [4, 5, 13].

A *nochange axiom*, written NCA, is the conjunction

$$\bigwedge_{p \in ATM} \neg O^p \rightarrow (p^I \leftrightarrow p^R).$$

Intuitively, the nochange axiom states that the value of each non-occluded atom persists from the initial state to the resulting state.

Given the language $\mathcal{L}_{stl}$, we write OCC to denote the set of all atoms of the form O^p.

We are now ready to state the minimization policy. Given an action scenario $\mathcal{A}$ we will use the following filtered preferential entailment policy:

$$NCA \wedge Circ[\mathcal{A}; OCC],$$

where $Circ[\Gamma; P]$ represents the standard second-order circumscription axiom which minimizes the predicates P relative to the finite formula Γ. In other words, we minimize the atoms of the form O^p relative to the action description conjoined with the initial state description, and then filter with the nochange axiom NCA. Since the predicates we minimize are actually 0-ary ones, $Circ[\mathcal{A}; OCC]$ is reducible to propositional logic. Consequently, any efficient decision procedure for classical propositional logic will suffice for computing preferential entailment.

Definition 4 (Preferential Entailment) Let $\mathcal{A}$ be an action scenario and suppose that $\alpha \in \mathcal{L}_{stl}$. We say that α is *preferentially entailed* from $\mathcal{A}$ in STL, written $\mathcal{A} |\!\approx_{stl} \alpha$, iff $NCA \wedge CIRC[\mathcal{A}; OCC] \models \alpha$, where $\models$ denotes the entailment relation of classical propositional logic. ■

5 Reformulating the MPMA in STL

In this section, we provide a reformulation of the MPMA in terms of STL.

Definition 5 Let $KB \in \mathcal{L}_{pma}$ and $\alpha \in \mathcal{L}_{pma}$ be a knowledge base and an update formula, respectively. An action scenario $\mathcal{A}$ *corresponding* to KB and α is given by

$$\mathcal{A} = KB^I \wedge \alpha^R \wedge Occlude(\alpha).$$

Example 7 Suppose that $KB = p$ and $\alpha = p \vee q$. The action scenario corresponding to KB and α is $p^I \wedge (p^R \vee q^R) \wedge O^p \wedge O^q$. ■

5.1 Proving Equivalence between MPMA and STL

Theorem 9 Assume that the language $\mathcal{L}_{pma}$ is based on a set of atoms ATM. Let $KB \in \mathcal{L}_{pma}$ and $\alpha \in \mathcal{L}_{pma}$ be a knowledge base and an update formula, respectively. Suppose that α is in Blake canonical form and let P be the set of all atoms from ATM occurring in α. If $\mathcal{A}$ is an action scenario corresponding to KB and α, then for any formula $\beta \in \mathcal{L}_{pma}$

$$KB \star^P \alpha \models \beta \quad \text{iff} \quad \mathcal{A} |\!\approx_{stl} \beta^R. ■$$

5.2 The Decision Procedure for MPMA

Theorem 9 provides the basis for a simple decision procedure for computing the MPMA using STL.[5]

Let $\mathcal{L}_{pma}$ and $\mathcal{L}_{stl}$ be the languages used to describe knowledge base update and action scenario queries, respectively. Given an MPMA update query

$$KB \star^P \alpha \models \beta,$$

where KB and α are formulas in $\mathcal{L}_{pma}$ and α is in Blake canonical form, we first translate the update $KB \star \alpha$ into a corresponding action scenario $\mathcal{A}$, and we translate the query β in $\mathcal{L}_{pma}$ into its correlate β' in $\mathcal{L}_{stl}$. We then solve the equivalent problem

$$NCA \wedge CIRC[\mathcal{A}; OCC] \models \beta'$$

using a decision procedure for classical propositional logic.

Example 8 Let $ATM = \{p, q\}$, $KB = p \wedge q$ and $\alpha = \neg q$. We use Theorem 9 to show that p is entailed by $KB \star^{\{q\}} \alpha$. An action scenario corresponding to KB and α is $\mathcal{A} = p^I \wedge q^I \wedge \neg q^R \wedge O^q$. It is easily checked that $CIRC[\mathcal{A}, OCC] \equiv p^I \wedge q^I \wedge \neg q^R \wedge O^q \wedge \neg O^p$ and so $NCA \wedge CIRC[\mathcal{A}, OCC] \equiv p^I \wedge q^I \wedge \neg q^R \wedge O^q \wedge \neg O^p \wedge (p^I \leftrightarrow p^R)$. Since $NCA \wedge CIRC[\mathcal{A}, OCC] \models p^R$, we conclude that $KB \star^{\{q\}} \alpha \models p$. ■

[5]For an on-line implementation of a restricted first-order version of STL, called TAL (Temporal Action Logic), see http://www.anton.ida.liu.se/vital/vital.html.

6 The MPMA and Integrity Constraints

In this section, we provide a means of extending the MPMA operator for integrity constraints. To deal with integrity constraints in the framework of the classical PMA, it has been proposed in Katsuno & Mendelzon [14] that one define such an extension by $KB \star (A \wedge IC)$, where $\star$ denotes the classical PMA operator and IC is the conjunction of the set of integrity constraints under consideration. Unfortunately, as has been observed by many researchers (see for instance Herzig [11], this solution is very problematic. To show this, we use a classical example in Herzig [11], originally due to Lifschitz [16].

Example 9 There are three atoms $sw1$, $sw2$ and l standing for "switch 1 is up", "switch 2 is up" and "light is on", respectively. The integrity constraint $IC = l \leftrightarrow (sw1 \leftrightarrow sw2)$. We would like to update the database $KB = l \wedge sw1 \wedge sw2$ with the update formula $\alpha = \neg sw1$. It is well known that describing the problem with the classical PMA as

$$KB \star (\alpha \wedge IC),$$

will not provide intuitive results. The new knowledge base has two models, w_1 and w_2, given by $\{\neg sw1, \neg l, sw2\}$ and $\{\neg sw1, l, \neg sw2\}$, respectively. The model w_2 is obviously an unintended one.

It is interesting to note that using this solution, but replacing the PMA with the MPMA, gives the same two models. ■

The problem which is well understood by now is that integrity constraints must include fluent dependency information in one form or another. In this paper, we shall employ *causal rules* as described in Gustafsson & Doherty [13]. These are expressions of the form

$$\alpha \gg \beta \tag{2}$$

where α and β are formulas, referred to as an *antecedent* and a *consequent* of the rule, respectively. A rule of the form (2) has the following intuitive interpretation:

(1) The formula $\alpha \rightarrow \beta$ holds in both the initial and the updated knowledge base.

(2) If $\neg\alpha$ held in the initial knowledge base and α holds in the updated knowledge base, then there is a cause for β to hold in the updated knowledge base.[6]

Recall that in the MPMA we minimize change, excluding from the minimization the atoms occurring in the update formula. In the MPMA with integrity constraints we, in addition, exclude from the minimization the atoms occurring in the consequents of *active* causal rules, where a rule is said to be active iff its antedecent changed its truth-value from *False* to *True* during the update.

Example 9 (continued) To encode causal information contained in IC, we introduce two rules: $(sw1 \leftrightarrow sw2) \gg l$ and $\neg(sw1 \leftrightarrow sw2) \gg \neg l$. Suppose we want to check whether $w_1 = \{\neg sw1, \neg l, sw2\}$ should be considered as a model of KB updated by $\neg sw1$ under the integrity constraints IC and the above causal rules. Denote by u the only model of $KB \wedge IC \wedge [(sw1 \leftrightarrow sw2) \rightarrow l] \wedge [\neg(sw1 \leftrightarrow sw2) \rightarrow \neg l]$, *i.e.* $u = \{l, sw1, sw2\}$.[7] Note that the antecedent of the rule $\neg(sw1 \leftrightarrow sw2) \gg \neg l$ has changed its value from *False* to *True* during the update. Therefore, we are allowed to ignore the atom l while minimizing change. The atom $sw1$ is also ignored since it occurs in the update formula. On the other hand, the atom $sw2$ has the same value in both u and w_1. Accordingly, w_1 is to be viewed as an intended model of the considered update. Consider now the model $w_2 = \{\neg sw1, l, \neg sw2\}$. It is readily checked that we are allowed to ignore $sw1$ and l while minimizing change. However, the models u and w_1 still differ in the atom $sw2$. Acordingly, w_2 should not be viewed as the intended model of the considered update. ■

We now formalize the above idea.

Definition 6 A causal rule $\alpha \gg \beta$ is said to be *active* wrt a pair of interpretations $\langle w, v\rangle$ iff the truth-values of α in w and u are *False* and *True*, respectively. ■

Definition 7 Let w and u be two interpretations and suppose that α is a formula in Blake canonical form. Assume further that $CR = \{\alpha_i \gg \beta_i\}$ is a set of causal rules such that all β_i's are in Blake canonical form. Let P be a set of atoms given by $P =$

$$atm(\alpha) \cup \bigcup\{atm(\beta_i) : \alpha_i \gg \beta_i \text{ is active wrt } \langle w, u\rangle\}.$$

The *distance between w and u wrt α and CR* is $DIST(w,u) - P$. ■

Definition 8 Let α and CR be as in Definition 7. The *update of an interpretation w by a set of interpretations V* wrt α and CR, written $w \star_{\alpha,CR} V$, is the set of those elements from V whose distance wrt α and CR is $\emptyset$. ■

Definition 9 Let α and CR be as in Definition 7. Let KB be a knowledge base and suppose that IC is a formula representing integrity constraints. Let U be the set of all models of $KB \wedge IC \wedge T(CR)$, where $T(CR)$ denotes the formula $\bigwedge_{i=1}^{n}(\alpha_i \rightarrow \beta_i)$ and let V be the set of all models of $\alpha \wedge IC \wedge T(CR)$. The *update* of KB by α wrt IC and CR, written $KB *_{IC,CR} \alpha$, is given by

$$KB *_{IC,CR} \alpha = \bigcup_{w \in KB \wedge IC \wedge T(CR)} w \star_{\alpha,CR} V. \blacksquare$$

6.1 STL and Integrity Constraints

To represent the MPMA with integrity constraints in STL, we first slightly generalize the notion of an action scenario.

Let $CR = \{\alpha_i \gg \beta_i : i = 1, \ldots, n\}$ be a set of causal rules. We write $Trans(CR)$ to denote the formula

$$\bigwedge_{i=1}^{n}(\alpha_i^I \rightarrow \beta_i^I) \wedge \bigwedge_{i=1}^{n}(\alpha_i^R \rightarrow \beta_i^R) \wedge$$
$$\bigwedge_{i=1}^{n}(\neg\alpha_i^I \wedge \alpha_i^R \rightarrow Occlude(\beta_i)).$$

Definition 10 An *action scenario under integrity constraints IC and a set of causal rules CR* is the conjunction $IC^I \wedge IC^R \wedge Trans(CR) \wedge \alpha \wedge \beta$, where α is an initial state description and β is an action description. ■

Definition 11 Let KB, α, IC and CR be a knowledge base, an update formula, a formula representing integrity constraints and a set of causal rules, respectively. An action scenario $\mathcal{A}$ *corresponding* to KB, α, IC and CR is given by

$$\mathcal{A} = KB^I \wedge IC^I \wedge \alpha^R \wedge Occlude(\alpha) \wedge IC^R \wedge Trans(CR).$$

■

[6] That is, if $\neg\alpha$ held in the initial knowledge base and α holds in the updated knowledge base, we are allowed to change the values of atoms occurring in β to guarantee that this formula holds in the updated knowledge base.

[7] If KB is a knowledge base, IC is a formula representing integrity constraints and the set of causal rules is $\cup_{i=1}^{n}\{\alpha_i \gg \beta_i\}$, then the original knowledge base should be considered not as KB, but rather as $KB \wedge IC \wedge \bigwedge_{i=1}^{n}(\alpha_i \rightarrow \beta_i)$.

Theorem 10 Let KB, α, IC and CR be a knowledge base, an update formula, a formula representing the set of integrity constraints and a set of causal rules, respectively. Suppose that α and all consequents of causal rules are in Blake canonical form. If $\mathcal{A}$ is an action scenario corresponding to KB, α, IC and CR, then for any formula $\beta \in \mathcal{L}_{pma}$

$$KB \star_{IC,CR} \alpha \models \beta \quad \text{iff} \quad \mathcal{A} |\!\approx_{stl} \beta^R. \blacksquare$$

Practically, the theorem shows that we have provided an alternative syntactic characterization and implementation of MPMA with integrity constraints in terms of STL. Since STL is a very restricted version of a richer temporal logic TAL [4, 13], it would be straightforward to provide additional generalizations to MPMA, based on intuitions from TAL.

7 Comparisons with Existing Work

In this section, we compare our approach with that of Herzig [11], whose work is most closely related to ours. Additional comparisons and observations are made with del Val and Shoham [24] and Sandewall [22] in Doherty *et al.* [6].

7.1 Herzig

Herzig [11] provides a sound and complete decision procedure for the PMA by constructing an equivalent presentation of the PMA in terms of conditional logic.[8] A syntactic characterization of an update $w \star U$ of an interpretation w by a set of interpretations U is provided in terms of a conditional operator $>$ where $A > C$ is read as a hypothetical update: "if the current database is updated with A then C follows". Herzig proves the following proposition:

Proposition 1 (Herzig [11]) *Let KB, A and C be classical. Then $KB \star A \models C$ iff $KB \models A > C$.* $\blacksquare$

Herzig then shows how the conditional $A > C$ can be rewritten to a classical formula using normal forming which together with the proposition above provides the decision procedure.

In Section 8 of his paper, Herzig sketches how one might deal with integrity constraints by introducing a new update operator which after careful analysis can be shown to be quite similar in spirit to the MPMA operator presented in this paper. Although similar in concept, Herzig's decision procedure is still based on translating update queries into conditionals and then translating via normal forming to a classical formula. In addition, instead of computing one PMA update, he is forced to compute 2^n classical PMA updates per query, where n is the number of atoms dependent on any of the atoms in the update formula.

[8] A relationship between the PMA and conditionals has also been studied in Fariñas del Cerro *et al.* [7, 8, 9] and Grahne [12].

To show the relation between the MPMA update operator $\star^P$ and Herzig's new update operator, written $\star_{IC,DEP}$, we will use Example 9 which is also used in Herzig [11], but with the atoms renamed.

In addition to the integrity constraint IC, one must also include fluent dependency information in one form or another. Herzig does this in the following manner: He first specifies a dependency function, DEP, from atoms to sets of atoms such that $p \in DEP(p)$, for each atom p. In addition, $DEP(p)$ may contain atoms other than p. Intuitively, $q \in DEP(p)$ means that updates concerning p may change the truth-value of q. For a formula A, $DEP(A)$ stands for $\bigcup_{[p \in atm(A)]} DEP(p)$.

In the example, $DEP(l) = \{l, sw1, sw2\}$, $DEP(sw1) = \{sw1, l\}$, and $DEP(sw2) = \{sw2, l\}$. The dependence function DEP encodes the change dependency inherent in a causal reading of IC and represents a fluent dependency graph.

An update operator $\star_{IC,DEP}$ under a set of integrity constraints IC and a dependence function DEP is then defined by

$$KB \star_{IC,DEP} A = \left(\bigvee_{B \in CTX(A)} (KB \star B) \right) \wedge IC \wedge A$$

where $\star$ is the classical PMA operator and $CTX(A)$ is $\{l_1 \wedge \cdots \wedge l_n : l_i = p_i \,\text{or}\, l_i = \neg p_i\}$ if $DEP(A) = \{p_1, \ldots, p_n\}$. So, for example, if $A = \neg sw1$ and the function DEP is as specified above where $DEP(sw1) = \{sw1, l\}$, then

$$CTX(sw1) = \{sw1 \wedge l, sw1 \wedge \neg l, \neg sw1 \wedge l, \neg sw1 \wedge \neg l\}.$$

This means that if the cardinality of $DEP(\alpha)$ is n then the cardinality of $CTX(\alpha)$ is 2^n.

The role of $CTX(A)$ is central to the approach and essentially does the job that the set P of atoms in the MPMA operator $\star^P$ does together with the eliminant $\exists P.KB$, or that the OCC atoms together with the NCA axiom plays in the translation to STL. Taking the disjunction of PMA updates of KB with each of the formulas in $CTX(sw1)$, has the effect of generating the same models as one would with the eliminant of KB where $P = \{sw1, l\}$. It can easily be shown that

$$KB \star_{IC,DEP} \neg sw1 \equiv KB \star_{IC,CR} \neg sw1,$$

where,

$$KB = l \wedge sw1 \wedge sw2,$$
$$IC = (sw1 \leftrightarrow sw2) \leftrightarrow l,$$

and

$$CR = \{[(sw1 \leftrightarrow sw2) \gg l], \\ [\neg(sw1 \leftrightarrow sw2) \gg \neg l]\},$$

or the translation into STL where,

$$\mathcal{A} = KB^I \wedge IC^I \wedge IC^R \wedge \neg sw1^R \wedge O^{sw1} \wedge Trans(CR).$$

The advantage of using the latter is that the decision procedure is much more efficient and that the dependency information included in DEP is implicit in the expansion of the abbreviation $\gg$.

Herzig's new update operator can be also used for the empty set of integrity constants by putting $IC = \top$. It turns out, that the operator $\star_{\top,DEP}$ is just the MPMA operator, with the set of variable atoms identified with the set of all the atoms occurring in the update formula.[9] More precisely:

Theorem 11 For each KB and A

$$KB \star_{\top,DEP} A \equiv KB \star^P (A)$$

where $P = atm(A)$. ■

8 Discussion

We have demonstrated the benefits of applying intuitions derived from research on action and change in the temporal reasoning community to the BR/U paradigm. The result is a generalization of the PMA to MPMA which handles disjunctive updates and integrity constraints interpreted as causal constraints. We have provided a number of syntactic characterizations of the MPMA and shown how an MPMA query can be mapped into a query of a temporal narrative represented in STL, a simple temporal logic for reasoning about action and change. Since STL is the propositional fragment of a highly expressive 1st-order temporal logic, it should be straightforward to generalize the results described here to the 1st-order case, where it has already been shown that the circumscription formula associated with the minimization policy in the full first-order version of STL is reducible to a first-order formula. An additional generalization would include the modeling of iterated belief update in MPMA in terms of narratives with more than two timepoints. Because the minimization policy associated with STL is similar to many other current approaches using temporal logics, we also believe that additional generalizations to the BR/U paradigm such as distinguishing between observations and action effects, or concurrent update can be added to MPMA or at least fully understood in this context.

[9]This means that Herzig's new update operator does not distinguish between redundant and non-redundant atoms occurring in the update formula. In consequence, it is syntax dependent and may lead to counterintuitive results (see Example 3).

Acknowledgements

Patrick Doherty and Witold Łukaszewicz have been partially supported by grants from the Wallenberg Foundation and the Swedish Research Council for Engineering Sciences (TFR). Ewa Madalińska-Bugaj is supported by grant KBN 8 T11C 035 111. Thanks to Jonas Kvarnström and Patrik Haslum for comments.

References

[1] G. Brewka, J. Hertzberg. How to Do Things with Worlds: on Formalizing Actions and Plans. *Journal of Logic and Computation*. Vol. 3, No. 5, 1993, 517-532.

[2] F. M. Brown. *Boolean Reasoning*. Kluwer Academic Publishers, 1990.

[3] M-O. Cordier, P. Siegel. A Temporal Revision Model for Reasoning about World Change. In *Int'l. Conference on Principles of Knowledge Representation and Reasoning*, 732-739, Morgan Kaufmann, 1992.

[4] P. Doherty. Reasoning about Action and Change Using Occlusion. In: *Proc. of the 11th European Conference on Artificial Intelligence*, 1994, 401-405.

[5] P. Doherty, W. Łukaszewicz. Circumscribing Features and Fluents. In: *Proc. of the 1st Int'l. Conference on Temporal Logic*, Lecture Notes on Artificial Intelligence, vol. **827**, Springer-Verlag, 1994, 82-100.

[6] P. Doherty, W. Łukaszewicz, E. Madalińska-Bugaj. The PMA and Relativizing Minimal Change for Action Update: Extended Report. *Linköping Electronic Articles in Computer and Information Science*, 1998. `http://www.ep.liu.se/ea/cis/`.

[7] L. Fariñas del Cerro, A. Herzig. An Automated Modal Logic for Elementary Changes. In: P. Smets, A. Mamdami, D. Dubois and H. Prade (eds.), *Non-Standard Logics for Automated Reasoning*, Academic Press, 1988, 63-79.

[8] L. Fariñas del Cerro, A. Herzig. A Conditional Logic for Updating in the Possible Model Approach. In: *Proc. of the 18th German Conference on Artificial Intelligence*, Springer-Verlag, LNAI, **861**, 1994.

[9] L. Fariñas del Cerro, A. Herzig. All Other Things Being Equal: On a Notion of Inertia in Conditional Logic. In: E. Ejerhed and S. Lindström (eds.), *Logic, Action and Cognition*, Kluwer Academic Press, to appear.

[10] N. Y. Foo, A. S. Rao. Minimal Change and Maximal Coherence: A Basis for Belief Revision and Reasoning about Actions. *In Proc. of the 11th Int'l. Joint Conference on Artificial Intelligence*, 1989.

[11] A. Herzig. The PMA Revisited. In: *Proc. of the 5th Int'l. Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann, 1996, 40-50.

[12] G. Grahne. Updates and Counterfactuals. In *Proc. of the 2nd Int'l. Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann, 1991, 269-276.

[13] J. Gustafsson, P. Doherty. Embracing Occlusion in Specifying the Indirect Effects of Actions. In: *Proc. of the 5th Int'l. Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann, 1996, 87-98.

[14] H. Katsuno, A. O. Mendelzon. On the Difference between Updating a knowledge base and revising it. In: *Proc. of the 2nd Int'l. Conference on Principles of Knowledge Representation and Reasoning*, KR'91, Morgan Kaufmann, 1991, 387-395.

[15] G.N. Kartha, V. Lifschitz. Actions with Indirect Effects (Preliminary Report). In *Proc. of the 4th Int'l. Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann, 1994.

[16] V. Lifschitz. Frames in the Space of Situations. *Artificial Intelligence J.*, **46**, 1990, 365-376.

[17] F. Lin. Embracing Causality in Specifying the Indirect Effects of Actions. *In Proc. of the 14th Int'l. Joint Conference on Artificial Intelligence*, 1995.

[18] F. Lin. Embracing Causality in Specifying the Indeterminate Effects of Actions. *In Proc. of the National Conference on Artificial Intelligence*, 1996.

[19] N. McCain, H. Turner. A Causal Theory of Ramifications and Qualifications. In: *Proc. of the 14th Int'l. Joint Conference on Artificial Intelligence*, Morgan Kaufmann, 1995, 1978-1984.

[20] P. Peppas, W. Wobcke. On the Use of Epistemic Entrenchment in Reasoning about Action. In: *Proc. of the 10th European Conference on Artificial Intelligence*, 1994, 403-407.

[21] E. Sandewall. *Features and Fluents. A Systematic Approach to the Representation of Knowledge about Dynamical Systems.* Volume 1. Oxford University Press, 1994.

[22] E. Sandewall. Assessments of Ramification Methods that use Static Domain Constraints. In *Proc. of the 5th Int'l. Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann, 1996.

[23] M. Thielscher. Ramification and causality. *Artificial Intelligence J.*, 1997.

[24] A. del Val, Y. Shoham. Deriving Properties of Belief Update from Theories of Actions (II). In *Proc. of the 13th Int'l. Joint Conference on Artificial Intelligence*, pages 732–737, 1993.

[25] M. Winslett. Reasoning about Action Using a Possible Model Approach. In: *Proc. AAAI-88*, St. Paul, MN, 1988, 89-93.

[26] M. Winslett. *Updating Logical Databases.* Cambridge Tracks in Theoretical Computer Science. Cambridge University Press, 1990.

[27] M. Winslett. Updating Logical Databases. In: D. Gabbay, A. Galton, J. A. Robinson (eds.). *Handbook of Logic in Artificial Intelligence and Logic Programming.* Vol. 4 (Epistemic and Temporal Reasoning), 1995, Oxford University Press, 133-174.

Quantifiers and Operations on Modalities and Contexts

Tom Costello
Computer Science Department
Stanford University
Stanford, CA 94305

Anna Patterson
Computer Science Department
Stanford University
Stanford, CA 94305

Abstract

We can reason about theories, as well as in them. Many natural phenomena can be captured by theories, often by introducing a modality, true just of the theory. Rather than treat these phenomena as the sentences true in this modality, we can treat the entire theory or modality as an object. This point of view was proposed by McCarthy, who proposed calling these reified modalities contexts.

Contexts, viewed as theories or modalities, have structural properties, and associated natural structural operations, such as union and intersection. These structural properties are most naturally captured by an algebra, reminiscent of dynamic algebra. This algebra has a set of operations that can be applied to contexts to form new contexts.

We introduce an algebra, reminiscent of dynamic algebra, but which acts on modalities, not relations. This algebra allows the construction of modalities from simpler modalities. We use the natural correspondence between a modality and the underlying accessibility relation to make this connection. Thus the operations on modalities are induced by operations on the underlying relations which respect the properties required of modalities.

It becomes possible then to have a larger space of modalities than those that are explicitly named. We add quantifiers that range over modalities, so that we can state propositions like, "no context with the property P exists", or "all contexts obey Q". We give an axiomatization of these quantifiers which we show is complete with respect to a natural semantics.

1 PUTTING MODALITIES TOGETHER

1.1 MODALITIES

Modal operators are widely used to capture intensional aspects of knowledge, belief, temporality and action. They capture the notion that there may be different viewpoints from which a particular phenomena can be seen. These different vantage points can be *agents beliefs*, *points in time*, *differing stages of knowledge*, or different *contexts*.

A unifying semantics for these notions was proposed by Kripke. The essential idea is that a new syntactic operator, $\Box$, is semantically modeled by a set of possible worlds, W, and an accessibility relation on worlds R. Each world is given an assignment to propositional atoms[1], and the truth condition for $\Box\phi$ at a world w, is defined as the truth of ϕ in all worlds accessible from w.

A natural extension of this idea is to allow more than one modality. One popular use of multi-modal systems is to have modalities which represent the beliefs of each member of a set of believers. (Halpern 1995) provides a recent overview of this approach. It is then customary to label each modal operator with a distinguishing subscript. If $\Box_{Sally}$ denotes the beliefs of Sally, and $\Box_{Harry}$, the beliefs of Harry, then if Sally believes that Harry believes that Sally likes Harry, we would write, $\Box_{Sally}\Box_{Harry}Likes(Sally, Harry)$.

[1]Alternately, each world can have an associated first order structure, or even a non-standard semantical structure, such as a three valued model. The choice of a propositional valuation is only for simplicity as most of these results extend readily to richer valuation functions.

1.2 MAKING NEW MODALITIES FROM OLD

In this paper, we ask the broad question "What modalities should exist" and "Given that we already have a certain set of modalities, what other modalities must also exist". Thus, we differ from previous work on indexed modalities in that we are interested in studying the structure of modalities themselves, not in stating facts that require multiple modalities for their expression.

We can imagine that we have the two modalities $\Box_{Sally}$ and $\Box_{Harry}$ that we considered earlier. It seems obvious that there is a set of beliefs that they share. This set of beliefs seems to be a well behaved modality. Other modalities are the union of the beliefs, the propositions they do not believe, and many others.

One way of posing the question of what other modalities might exist, is to ask "What would a system that quantified over modalities look like?" We first present a simple system that allows us to quantify over modalities, but gives us no way to generate these modalities. We then look at natural ways to generate modalities, using some notions from dynamic algebra.

1.3 IDENTITY CONDITIONS ON MODALITIES

As well as a theory of what modalities exist, we also need an identity criterion for modalities. We take the most obvious criterion, *extensionality.* That is, we consider two modalities equal if they agree on all sentences.

$$c = c' \overset{def}{=} \text{ for all } \phi . \Box_c \phi \equiv \Box_{c'} \phi$$

We consider what structural properties this decision has on the underlying relations. We show it identifies all *bisimilar* models.

1.4 PLAN OF PAPER

We first look at Kripke models, which provide the basic model theory of modalities.

We give a sound and complete axiomatization for arbitrary universes of relations. This gives a simple system which allows quantification over modalities, but which does not have any closure conditions on what modalities should exist.

We then suggest three operators that make new modalities from old modalities. Therefore our universe of modalities will include all modalities definable using these operations. We give an axiomatization of these operators that we show is sound and complete for universes that include all *first order definable functions on modalities.* This shows that these three operators are sufficient to define all first order definable functions on modalities.

To prove this theorem we have to characterize all *first order definable functions on modalities.* We consider when the modalities Kripke models define are equivalent. This needs the notion of bisimulation. We then look at what functions on the relations in Kripke models give rise to functions on modalities. We then prove a theorem which characterizes the first order definable functions on relations that give rise to functions on modalities. We show that the syntactic class of first order definable functions on modalities can be captured by our three operations on modalities.

We conclude by looking at previous related work in AI, and suggesting some natural uses for the operators we have introduced.

2 MODALITIES AND RELATIONS

We now state the standard definitions of Kripke structures, and review how Kripke models can give a semantics to a normal modality.

2.1 KRIPKE MODELS

We define propositional modal languages in the usual way.

Definition 1 *A propositional modal language over a set P of propositional atoms is the smallest set of formulas containing each element of P, and $\neg\varphi$, $\varphi \wedge \psi$, $\Box\varphi$, where ϕ and ψ are formulas.*

We can define the other propositional connectives, $\vee$, $\rightarrow$, $\equiv$, and the modal operator $\Diamond\phi = \neg\Box\neg\phi$ in the usual way.

Definition 2 *A Kripke structure for propositional modal logic over a set P of propositional atoms is a triple $\langle W, R, V\rangle$ where W is a non-empty set (the set of* worlds*), R is a binary relation on W and V is a function assigning to every world a set of propositional atoms from P.*

Definition 3 *The satisfaction relation $\models$ for $K = \langle W, R, V\rangle$, a Kripke structure over a set of propositional atoms P is a relation between worlds and modal formulas. We define this relation inductively over the* *structure of a formula. Thus, for any $w \in W$,*

$$\begin{array}{lcr} w,K \models p & \equiv & p \in V(w) \\ w,K \models \neg\varphi & \equiv & w,K \not\models \varphi \\ w,K \models \varphi \wedge \psi & \equiv & w,K \models \varphi \textit{ and } w,K \models \psi \\ w,K \models \Box\varphi & \equiv & \forall w' \in W. wRw' \textit{ implies } w',K \models \varphi \end{array}$$

For the definition of negation we read, $w, K \not\models \varphi$ as "it is not the case that $w, K \models \varphi$".

Definition 4 *Let $K = \langle W, R, V\rangle$ be a Kripke structure, then a formula φ is* satisfied *in K if and only if $\exists w \in W.\ \ w, K \models \varphi$. A formula φ is* valid in K, *written $K \models \phi$, if and only if $\forall w \in W.\ \ w, K \models \varphi$. A formula φ is* valid *if and only if φ is valid in all Kripke structures.*

3 QUANTIFICATIONAL MODAL LOGIC

We look at quantifying over modalities in a modal logic. For simplicity, we start with the propositional case of the modal logic K^2 with multiple modalities. We extend multi-modal K to also allow modal variables, which we write as subscripts, such as $\Diamond_x$. We write modal constants normally, as $\Diamond_c$, for the constant c. We then extend this language to consider quantifiers over modalities or contexts. We do not allow quantification over objects as our language is propositional. We also do not allow quantification over propositions as our language is not second order in this respect.

Later we use an alternate notation, $ist(c, \phi)$, where c plays the role of a modal constant or subscript. This allows a more traditional syntax in some respects.

3.0.1 Syntax

The *symbols* of our language, denoted $L_{\mathbb{C},\mathbb{P}}$ include a countable set of propositional letters $\mathbb{P}$, a countable set of modal constants $\mathbb{C}$, and a countable set of modal variables which we denote by x_j and y_j where $j \in \omega$. We also include symbols for the usual classical connectives, $\wedge$ and $\neg$, the modal symbol $\Box$, parentheses) and (, and lastly the quantifier symbol $\forall$. We use $\phi \rightarrow \psi$ as an abbreviation for $\neg(\phi \wedge \neg\psi)$ as usual.

The set of formulas of our language is the smallest set containing

- All propositional letters, $\mathbb{P}$.

[2]classical propositional logic plus the modality $\Box$ with modal distribution and necessitation

- If ϕ and ψ are formulas, then $(\phi \wedge \psi)$ and $(\neg\psi)$ are formulas.
- If ϕ is a formula, and i is a modal constant, or a modal variable, then $(\Box_i\phi)$ is a formula.
- If ϕ is a formula, and x is a modal variable, then $(\forall x.\phi)$ is a formula.

In the above definition, the formulas ϕ and ψ are subformulas of the resulting formula, furthermore, the sub-formula property is transitive. We say an occurrence of a variable x is bound in a formula ϕ, if it is contained in a sub-formula of ϕ, of the form $\forall x.\psi$. We assume variables are renamed to avoid clashes in the usual way. We say an occurrence of a variable x is free in a formula if it is not bound. We call a formula with no free variables a sentence.

3.1 SEMANTICS

The usual semantics of modal logic we just presented and is based around a set of worlds, and a binary relations over worlds. For multi-modal models we need a set of binary relations over worlds, one for every modal constant. Hence every modality is associated with a relation, which we call its accessibility relation. We also have a valuation function, that is a function that assigns a set of propositional atoms to each world.

Definition 5 *A structure for propositional quantified modal logic over a set P of propositional atoms is a quadruple $\langle W, U, V, A\rangle$ where W is a non-empty set (the set of* worlds*), U is a non-empty universe of binary relations on W and V is a function assigning to every world a set of propositional atoms from P and A is an assignment function from modal variables and constants to elements of U.*

We often denote the accessibility relation of A applied to constant c, as A_c, similarly A_x for modal variable x. We write A_i to mean that i is either a modal constant or variable.

The semantics of quantified modal logic, i.e. our language $L_{\mathbb{C},\mathbb{P}}$, is defined as follows.

Definition 6 *The satisfaction relation $\models$ for $S = \langle W, U, V, A\rangle$, a structure for quantified modal logic over a set of propositional atoms P is a relation between worlds and quantified modal formulas. We define this relation inductively over the structure of a formula.*

Thus, for any $w \in W$,

$$\begin{array}{lcl} w,S \models_{\forall} p & \equiv & p \in V(w) \\ w,S \models_{\forall} \neg\phi & \equiv & w,S \not\models_{\forall} \phi \\ w,S \models_{\forall} \phi \wedge \psi & \equiv & w,S \models_{\forall} \phi \text{ and } w,S \models_{\forall} \psi \\ w,S \models_{\forall} \Box_i\phi & \equiv & \forall w'.wA_iw' \text{ implies } w',S \models_{\forall} \phi \\ w,S \models_{\forall} \forall x.\phi & \equiv & \forall R \in U.w, S[A[R/x]/A] \models_{\forall} \phi \end{array}$$

where $S[A[R/x]/S]$ is the structure S except that A has been replaced with $A[R/x]$ where $A[R/x]$ is like A with the possible difference that it assigns the relation R to the variable x.

Definition 4 lifts naturally to our definition of satisfaction in the quantified modal logic setting. For the remainder we use those terms as defined.

We have placed no restrictions on the universe of relations U. We now ask the question, what relations should be included in the universe U.

3.2 JUST CONSTANTS

We can follow the lead of first order logic by considering all non-empty domains of relations possible. For quantified modal logic this is not an appropriate long-term solution because our domains should satisfy certain closure conditions. That is a relation can be constructed from other relations using natural operations that we define in Section 3.7 such as union. This means that we can construct a relation by unioning two relations in the universe but find that its union does not exist in the universe. We would like our universes to be closed under all appropriate operations, but first we need to study the simple case of universes and determine what are the "appropriate operations" for combining modalities.

As a starting point, we give an axiomatization of the case where there is no restriction placed upon the universe of relations.

3.2.1 Axiomatics

The following are the axioms and rules of our minimal system, which we call $\forall$. We assume a Gentzen style system for concreteness.

All valid principles of propositional logic

$\vdash \Box_i(\phi \to \psi) \to (\Box_i\phi \to \Box_i\psi)$ Modal Distribution

$\dfrac{\vdash \phi}{\vdash \Box_i\phi}$ Necessitation

$\dfrac{\Gamma \vdash \psi}{\Gamma \vdash \forall x.\psi}$
provided that x does not occur free in Γ

$\dfrac{\Gamma \vdash \forall x.\psi}{\Gamma \vdash \psi[i/x]}$ where $\psi[i/x]$ is the result of substituting each free occurrence of x by i

We define logical derivation in the standard way, and denote it $\vdash_{\forall}$. We say $\Gamma \models_{\forall} \phi$, read, ϕ is a valid consequence of Γ, if, for all models $S = \langle W, U, V, A \rangle$ and for all worlds w in W, that satisfy each formula in Γ, it is the case that $w, S \models_{\forall} \phi$. We now show soundness and completeness for this basic case.

Theorem 7 *If ϕ is a sentence in $L_{\mathbb{C},\mathbb{P}}$, then $\vdash_{\forall} \phi$ implies $\models_{\forall} \phi$.*

Proof: We show that the axioms are satisfied in all models and that the rules of inference preserve validity. Let S be an arbitrary quantified modal logic model i.e. some $S = \langle W, U, V, A \rangle$.

We have one non-propositional axiom.

- $\Box_i(\phi \to \psi) \to (\Box_i\phi \to \Box_i\psi)$. We show this is satisfied in all models, by deriving the right-hand side from the definition of the left hand side and the definition of $\models_{\forall}$.

  First assume that S models $\Box_i(\phi \to \psi)$ which expands to $w, S \models_{\forall} \Box_i(\phi \to \psi) \equiv$
  $\forall w'.wA_iw'$ implies $w', S \models_{\forall} (\phi \to \psi)$
  By the definition of entailment of $\to$ this gives,
  $\forall w'.wA_iw'$ implies
  $(w', S \models_{\forall} \phi$ implies $w', S \models_{\forall} \psi)$
  We now assume the left part i.e.
  $\forall w'.wA_iw'$ implies $w', S \models_{\forall} \phi$
  to obtain $\forall w'.wA_iw'$ implies $w', S \models_{\forall} \psi$
  Discharging our assumptions gives
  $w, S \models_{\forall} \Box_i(\phi \to \psi) \to$
  $(\forall w'.wA_iw'$ implies $w', S \models_{\forall} \phi) \to$
  $(\forall w'.wA_iw'$ implies $w', S \models_{\forall} \psi)$
  Which by the definition of $\to$ and $\Box$ gives
  $w, S \models_{\forall} \Box_i(\phi \to \psi) \to w, S \models_{\forall} \Box_i\phi \to \Box_i\psi$.

We have two quantifier rules.

- Our inductive assumption gives us that $\Gamma \models_{\forall} \psi$. Hence, in all models S where Γ is satisfied, ψ is satisfied. We also given that x does not occur in Γ; hence the A_x component of any satisfying world in S is never referenced when satisfying Γ. Therefore any A_x will suffice. This is just the condition under which $\Gamma \models_{\forall} \forall x.\psi$.
- Our inductive assumption gives us that $\Gamma \models_{\forall} \forall x.\psi$. Hence, in all models S the worlds w where Γ is satisfied, $\forall x.\psi$ is satisfied. The definition of satisfaction for $\forall$ gives us that $w, S \models_{\forall} \Gamma$ implies $\forall R \in U.w, S \models_{\forall} \psi$. But we note that this implies that in particular the interpretation of i, can be used, giving. $w, S[A[i/x]/A] \models_{\forall} \psi$. This is equivalent to $w, S \models \psi[i/x]$, for each free occurrence of x, as required.

We now show the other rules preserve validity. We assume the assumptions logically entail the premises and derive the conclusion from the assumptions.

- Propositional rules. Immediate.
- **Nec:** If ϕ is true without assumptions, then it is true in all world/model pairs. Therefore, it is true in all worlds in a particular model. In that model, we have $\Box_i \phi$, as it is true in all i accessible worlds.

Theorem 8 *If ϕ is a sentence in $L_{\mathbb{C},\mathbb{P}}$ and Γ is a set of sentences of $L_{\mathbb{C},\mathbb{P}}$, where $\Gamma \models_{\forall} \phi$, then $\Gamma \vdash_{\forall} \phi$*

Proof: It suffices to show that every consistent set Σ of sentences has a model. We take any consistent set of sentences Σ, and extend it to a maximal consistent set of sentences Σ'.

We then add a countable number of new modal constants $C' = \{c'_n | n \in \omega\}$, as witnesses for every existential. A countable number suffices as our language $L_{\mathbb{C},\mathbb{P}}$ is countable. This gives the new set of sentences, Σ'', in the language $L_{\mathbb{C}\cup C',\mathbb{P}}$. We note that Σ'' is a maximal consistent set, as we chose each constant to be a witness for an existential that was already completely defined.

We now construct a model $M = \langle W, U, V, A\rangle$, from this set of sentences Σ'' and show that this model contains a world w that satisfies all of the sentences in Σ.

First we construct a function I from which M will be defined:

Let s, s' denote finite sequences of modal constants and denote concatenation by $\cdot$. The empty sequence is denoted ϵ and the singleton sequence is denoted by its element.

We now define I'.

- $I'(\epsilon) = \Sigma''$
- $I'(c) = \{\varphi \mid \Box_c \varphi \in \Sigma''\}$ where $c \in \mathbb{C} \cup C'$.
- $I'(c \cdot s') = \{\varphi \mid \Box_c \varphi \in I'(s')\}$ where $c \in \mathbb{C} \cup C'$

We can associate with a set of sentences Δ its set of maximal consistent extensions of Δ. Thus $I(s)$ returns this set of maximal consistent extensions of $I'(s)$. I and I' are deeply related for $\varphi \in I'(s)$ if and only if $\forall w.w \in I(s)$ implies $\varphi \in w$. We need this correspondence when appealing to the construction of M in the completeness proof.

From left to right we only use that $I(s)$ is the maximal consistent extensions of s. From right to left we use the property of intersection and maximal consistency to conclude that if $I'(s)$ is a theory (i.e. logically closed) then it is equal to the intersection of its maximal consistent extensions. To show that $I'(s)$ is a theory we need that it only obeys necessitation and distribution. For this we use induction and the base case that Σ'' is closed under these rules.

Construction of W: We take as our set of worlds W, the union of the range of I. This means each world is a maximal consistent set of sentences.

Construction of A: We construct A_c for each $c \in \mathbb{C} \cup C'$ as the relation, $\{\langle w, w'\rangle | \exists s.w \in I(s) \wedge w' \in I(c \cdot s)\}$.

Construction of U: We take as our universe of relations $U = \{A_c | c \in \mathbb{C} \cup C'\}$.

Construction of V: $V(w)$ if and only if $p \in w$.

At this juncture technically, we must show M is a quantified modal model by showing that W and U are non-empty. W is non-empty by construction and that U is non-empty follows from the non-emptiness of W.

Finally we need to show that the restriction of this model $M = \langle W, V, U, A\rangle$, to the constants in $\mathbb{C}$ is a model of Σ.

We show this by induction on the structure of sentences in Σ''. We show that every set of sentences w in $I(s)$, is satisfied in M at w by showing that $\forall w.\forall \gamma.[\gamma \in w$ if and only if $w, M \models_{\forall} \gamma]$. As we are doing our induction on sentences, rather than formulas, we will need to use the fact that we introduced sufficient witnesses so that every existential had a witness.

- $\gamma = p$. We immediately get that $p \in w$ if and only if $w, M \models_\forall p$ because $w, M \models_\forall p$ if and only if $p \in V(w)$ and our structure is defined such that $p \in w$ if and only if $p \in V(w)$.

- $\gamma = \phi \wedge \psi$. We have by assumption that $\phi \in w$ if and only if $w, M \models_\forall \phi$ and $\psi \in w$ if and only if $w, M \models_\forall \psi$. Since w is a set of maximal consistent set of sentences then $\phi \wedge \psi$ are in w and by definition of $\models_\forall$ and the hypotheses we have that $w, M \models_\forall \phi \wedge \psi$.

- $\gamma = \neg\phi$. We have that $\phi \notin w$ if and only if $w, M \not\models_\forall \phi$. By the definition of satisfaction in a model $w, M \models_\forall \neg\phi$ and because w is maximal consistent $\neg\phi \in w$.

- $\gamma = \Box_c\phi$. We need that $\Box_c\phi \in w$ if and only if $w, M \models_\forall \Box_c\phi$. We assume that $\phi \in w$ if and only if $w, M \models_\forall \phi$.

  The satisfaction condition for $w, M \models_\forall \Box_c\phi$ is $\forall w'.wA_cw'$ implies $w'M \models_\forall \phi$.

  Let s be any sequence such that $w \in I(s)$. Then, consider $s' = c \cdot s$. By construction $\phi \in w' \in I(s')$ if and only if wA_cw', therefore all accessible worlds model ϕ, hence $w, M \models_\forall \Box_c\phi$. Our induction hypothesis gives us that all accessible worlds w' have ϕ as an element.

  Now $\Box_c\phi$ must be in w; if it were not then $\neg\Box_c\phi$ would be in w as w is maximal consistent. However, we have shown that $\phi \in w'$ for all accessible w' hence $\neg\Box_c\phi$ cannot be in w hence $\Box_c\phi$ must be in w.

- $\gamma = \forall x.\phi$. We need that $\forall x.\phi \in w$ if and only if $w, M \models_\forall \forall x.\phi$. We assume that $\phi \in w$ if and only if $w, M \models_\forall \phi$.

  The satisfaction condition for $w, M \models_\forall \forall x.\phi$ is that $\forall R \in U.M[A[R/x]/A] \models_\forall \phi$.

  As w is complete and consistent if $\forall x.\phi \in w$ and $\neg\phi[c/x] \in w$ for some c, then we could derive a contradiction. Since w contains $\phi[c/x]$ for each $c \in \mathbb{C} \cup C'$, we can apply our induction hypothesis and the definition of satisfaction in a model to conclude that $\forall x.\phi \in w$ implies $w, M \models_\forall \forall x.\phi$.

  If $w, M \models_\forall \forall x.\phi$ then have $\forall R \in U.M[A[R/x]/A] \models_\forall \phi$ whence we apply our induction hypothesis and consistency to conclude that $\forall x.\phi \in w$.

$\Sigma'' \supset \Sigma$ hence M above suffices. ∎

3.3 ALL UN-NESTED MODAL FORMULAS

We now return to our question. What should the universe of modalities look like. In order to determine the correct closure conditions, we would like to know something about possible operations under which the universe should be closed.

We are going to consider what operations on modalities should exist.

3.3.1 Test

A simple way of making modalities is to make a modality from a modal formula.

If B is a modal propositional formula without modalities then we can define $\Diamond_{i:B?}\phi$, as

$$\Diamond_{i:B?}\phi \equiv (\Diamond_i\phi) \wedge B \tag{1}$$

Before we get carried away defining operations, we must ensure that each operation actually satisfies the properties of a modality.

We show that this **Test** operator always defines a modality. We must show that we do not imply that $\Diamond_{i:B?}\bot$, as this denies a theorem that for all i $\Box_i\top$. If we substitute $\bot$ for ϕ in the above we see that so long as $\Diamond_i\bot$ is false, $\Diamond_{i:B?}\bot$ will be false. We also get that the boxed form of the operation obeys necessitation and modal distribution which can be checked simply by expanding the definitions.

3.3.2 Union

Other ways of defining new modalities suggest themselves. Perhaps the most natural of these is to allow the definition of a new modality in terms of combining two old modalities via union.

Thus, given modalities, $\Diamond_1$ and $\Diamond_2$, we can allow ourselves to make their union as follows,

$$\Diamond_{1\cup 2}\phi \equiv \Diamond_1\phi \vee \Diamond_2\phi \tag{2}$$

Again due to a similar argument $\Diamond_{1\cup 2}\phi$ will not be false unless one of the original modalities is false. The boxed form of this union operation obeys necessitation and modal distribution. This can be checked simply by expanding the definitions.

3.3.3 Intersection and Negation?

However, it is worthwhile noticing that

$$\Diamond_{1\cap 2}\phi \equiv \Diamond_1\phi \wedge \Diamond_2\phi$$

does not define a new modality, nor does

$$\Diamond_{-1}\phi \equiv \neg\Diamond_1\phi.$$

The second can be shown by noting that $\neg\Diamond_1\neg(p \vee \neg p)$ is equivalent to $\Box_1 p \vee \neg p$, which is valid by necessitation. Therefore $\Diamond_{-1}\neg(p \vee \neg p)$ is true for all modalities $\Diamond_1$, but we have that $\neg\Diamond_{-1}\neg(p \vee \neg p)$ — a contradiction. Therefore adding the above definition would render the system inconsistent.

The definition of $\Diamond_{1\cap 2}$ does not define a modality unless $\Diamond_1\phi \to \Diamond_2\phi$ or vice-versa. In this case it yields the stronger. If there is a ϕ such that $\Diamond_1\phi \wedge \Diamond_2\neg\phi$, then we have $\Diamond_{1\cap 2}\phi \vee \neg\phi$ but not $\Diamond_{1\cap 2}\phi$ or $\Diamond_{1\cap 2}\neg\phi$, which is inconsistent with modal distribution.

3.4 NESTED BELIEFS

Another natural operation to add is to allow nested modalities in the defining formula. Thus we could allow the definition of a modality corresponding to Sally's opinion of Harry's beliefs.

3.4.1 Composition

$$\Diamond_{x;y}\phi \equiv \Diamond_y\Diamond_x\phi \tag{3}$$

Again due to a similar argument $\Diamond_{x;y}\phi$ will not be false unless one of the original modalities is false. The boxed form of this union operation obeys necessitation and modal distribution. This can be checked simply by expanding the definitions.

3.5 OPERATIONS AND UNIVERSES

Adding these operations means that universes for quantified modal logic would have to be closed under these operations. For if not then we might have two modalities $\Diamond_y$ and $\Diamond_x$ without their union or composition.

Therefore we now consider a system which corresponds to the universe of relations being closed under union, intersection with the worlds that satisfy a property, and relational composition i.e. composition of two binary relations R and S, is defined as , $\forall x\ y.S; R(x,y) \equiv \exists z.R(x,z) \wedge S(z,y)$.

We call the derivability relation with these new axioms 1, 2 and 3 $\vdash_{\mathbf{V}}$.

Definition 9 *We extend the satisfaction relation $\models_{\mathbf{V}}$ for $S = \langle W, U, V, A\rangle$, a structure for quantified modal logic inductively over the structure of the additional formulae and denote the extension $\models_{\mathbf{V}}$. Thus, for any $w \in W$,*

$$\begin{array}{lcl} w, S \models_{\mathbf{V}} \Diamond_{i:p?} & \equiv & w, S \models_{\mathbf{V}} \Diamond_i\phi \wedge p \\ w, S \models_{\mathbf{V}} \Diamond_{x\cup y} & \equiv & w, S \models_{\mathbf{V}} \Diamond_x\phi \text{ or } w, S \models_{\mathbf{V}} \Diamond_y\phi \\ w, S \models_{\mathbf{V}} \Diamond_{x;y} & \equiv & w, S \models_{\mathbf{V}} \Diamond_y\Diamond_x\phi \end{array}$$

Definition 10 *A quantified modal model $\mathbb{M} = \langle W, U, V, A\rangle$ is* closed *if the set U is closed under all first order definable functions f on modalities.*

A function $f: 2^{(W\times W)} \to 2^{(W\times W)}$ is a first-order definable function on modalities if there is some function g such that $g(\langle W, R, V\rangle) = \langle W, f(R), V\rangle$.

The annotation about the induced function g is that f must not be a relation, nor must it must take Kripke structures with the same theory to Kripke structures with a different theory. We saw before that intersection of modalities (although first-order definable) is not a function on modalities.

Definition 11 *We say that ϕ is valid under $\models_{\mathbf{V}}$, denoted $\models_{\mathbf{V}} \phi$, if for all closed quantified modal models $\mathbb{M} = \langle W, U, V, A\rangle$ and all worlds $w \in W$, $w, \mathbb{M} \models_{\mathbf{V}} \phi$.*

Theorem 12 *If ϕ is a sentence in $L_{\mathbb{C},\mathbb{P}}$, then $\vdash_{\mathbf{V}} \phi$ implies ϕ is valid under $\models_{\mathbf{V}}$.*

Proof Outline: We need to check the new axioms. The axioms involving modal sentences can be seen to be true by considering the truth conditions. ∎

Theorem 13 *If ϕ is a sentence in $L_{\mathbb{C},P}$ and $\models_{\mathbf{V}} \phi$, then $\vdash_{\mathbf{V}} \phi$*

The proof relies on some technical material about what constitutes a function on modalities that is presented in the next section. In particular we need to show that every first order definable function on modalities is representable by our three operations, union, composition and $i: B?$.

Proof: We add sufficient constants, with definitions, so that we have a constant corresponding to every every point in the free algebra of the introduced operations. We need to show that each of these constants is correctly interpreted. That is we need to show that the function I we defined in the proof of Theorem 8 correctly models sequences containing the newly defined modal terms. We show this by showing that we can find a set of worlds which models the modality.

We assign these by induction on the terms i.

- $i = (j: B?) \cdot s$ This is assigned the set, $I(j \cdot s) \cap \{w' | B \in w' \in W\}$.

- $i = (j \cup k)\cdot s$ This is assigned $I(i) = I(j\cdot s) \cup I(k\cdot s)$.
- $i = (j; k) \cdot s$ This is assigned $I(i) = I(k \cdot (j \cdot s))$.

Now that we have interpreted all the new constants, we need to check that they are interpreted correctly. This follows immediately from their construction.

We then need to show that this structure we have defined is closed under first order definable functions on modalities.

We use Corollary 18 in the next section.

By construction all modalities constructed from $\cup$, $c\colon B?$ and $c; c'$ are in our structure, thus Corollary 18 lets us conclude that our structure is closed under all first order definable functions on modalities.

The rest of the proof follows as before. ∎

3.6 BISIMULATION

We now introduce a notion, *bisimulation*, that we will use to define equality of modalities. In order to relate the syntactic question of when two modalities are true of the same formulas to the semantic question of a relation between accessibility relations we consider *bisimulation.*

Van Benthem showed the close connection between bisimulation (or more precisely *p-morphisms*) and definability in modal logics (1983). Using the classical transformation of modal formulas to first order formulas, he showed that a first order formula is preserved under bisimulation exactly when it is equivalent to the translation of a modal formula. We use this result to determine when the two Kripke models are extensionally equal. A key part of checking that a function on relations lifts to a function on modalities is recognizing that equal modal structures are sent to equal modal structures.

Definition 14 *Let* $K = \langle W, R, V\rangle$ *and* $K' = \langle W', R', V'\rangle$ *be two Kripke structures over propositional atoms* P*, then* K *is* bisimilar *to* K', $K \leftrightarrow K'$, *if there is a relation* $S \subseteq W \times W'$ *such that*

- $\forall w \in W\ \exists w' \in W'$ *such that* $w\ S\ w'$, *and*
- $\forall w' \in W'\ \exists w \in W$ *such that* $w\ S\ w'$, *and*
- $x\ S\ y$ *implies* $V(x) = V'(y)$, *and*
- $x\ S\ y$ *implies* $\forall x' \in W.\ x\,R\,x' \rightarrow (\exists y' \in W'.\ y\,R'\,y' \wedge x'\ S\ y')$, *and*
- $x\ S\ y$ *implies* $\forall y' \in W'.\ y\,R'\,y' \rightarrow (\exists x' \in W.\ x\,R\,x' \wedge x'\ S\ y')$.

This notion of bisimulation captures exactly that two models agree on all modal sentences.

Theorem 15 ((Benthem 1983)) *Two Kripke structures* M *and* N *are bisimilar,* $M \leftrightarrow N$ *if and only if*

($M \models \phi$ *if and only if* $N \models \phi$*) for all modal formulas* ϕ*.*

We use this theorem in the next section as our identity condition on first order relations. We need the identity condition on relations that corresponds to extensional equality on modalities, as we wish to consider what functions on modalities exists, in terms of their characterization as functions on relations.

Thus Kripke structures agree on all modal formulas if and only if they are bisimilar. Thus any function on from Kripke structures to Kripke structures is a function on modalities if and only if it maps bisimilar structures to bisimilar structures.

3.7 FIRST ORDER TRANSLATIONS

The propositional modal language we have introduced may be embedded into first order predicate logic, by the following *standard translation*, where $\varphi[y/x]$ is the result of replacing all free occurrences of x by y in φ.

$$\begin{array}{lcl}
(p)^* & = & Px \\
(\neg\varphi)^* & = & \neg(\varphi)^* \\
(\varphi \wedge \psi)^* & = & (\varphi)^* \wedge (\psi)^* \\
(\Box\varphi)^* & = & \forall y(R(x,y) \rightarrow (\varphi)^*[y/x]) \\
\text{for some} & & \text{new variable } y
\end{array}$$

In fact, only two variables are needed. We then have the following theorem.

Theorem 16 *[(Benthem 1983)] A first order formula is (equivalent to) a translation of a modal formula if and only if it is invariant for bisimulation.*

In the next section we use this theorem to show that our new operators can define all first order modalities.

3.8 DEFINABLE RELATIONS

We now ask what definable functions f on relations give rise to functions on modalities $g(\langle W, R, V\rangle) = \langle W, f(R), V\rangle)$. If we define a new Kripke model, by generating a new binary relation on worlds, when does this definition map Kripke models that have the same theory to Kripke models with the same theory?

The answer to this can be derived from the previous theorem—the only formulas that are invariant for bisimulation are those that are equivalent to modal formulas. In this case it suffices to be able to find a formula $\psi[q]$ such that $\langle W, R', V\rangle \models \Diamond\phi$ if and only if $\langle W, R, V\rangle \models \psi[\phi/q]$.

Theorem 17 *Let $K = \langle W, R, V\rangle$ be a Kripke structure over propositional atoms $\{p_1, \dots p_n\}$.*

Let $v: W \to 2^{\{p_1, \dots p_n\}}$ range over valuation functions and let $P_1, \dots, P_m$ be unary predicates on W, such that $(p_i)^* = P_i(x)$.

For all $S \subseteq W \times W$,

$$\forall v.\langle W, S, v\rangle \leftrightarroweq \langle W, R, v\rangle \text{ implies } \forall v.\langle W, f(R), v\rangle \leftrightarroweq \langle W, f(S), v\rangle,$$

if and only if, there is a modal formula ψ in the modal language with atoms $p_1, \dots, p_n, q$, such that

$$(\psi(q))^* \equiv \exists y.f(R)(x, y) \wedge Q(y)$$

where q is a new propositional letter, with $(q)^* = Q(x)$ and $\equiv$ is equivalence in first-order classical logic.

In fact, ψ can be chosen so that every occurrence of q in $\psi(q)$ is scoped by an even number of negations, and $\wedge$ does not occur outside any pair of modalities that bind q.

Proof: $\Leftarrow$ We are given an equivalence, and we need to show that a definition f preserves bisimulation. To show that $\langle W, f(R), V\rangle \leftrightarroweq \langle W, f(S), V\rangle$ it suffices to show that they agree on all modal formulas. We prove this by induction on formulas. As we have proved the equivalence for arbitrary q we can use it for arbitrary formulas φ. This gives us truth conditions for statements of the form $\langle W, f(R), V\rangle \models \Diamond\varphi$ in terms of $\langle W, R, V\rangle \models \psi[\varphi/q]$. Thus we can reduce the truth of all formulas in $\langle W, f(R), V\rangle$ to the truth of formulas in $\langle W, R, V\rangle$. We can similarly reduce the truth of all formulas in $\langle W, f(S), V\rangle$ to the truth of the same formulas in $\langle W, S, V\rangle$. As by assumption if $\langle W, R, V\rangle \leftrightarroweq \langle W, S, V\rangle$ then these models agree on all formulas. Therefore the new models agree, and are thus bisimilar, as required.

$\Rightarrow$ We are given that f preserves bisimulations and need to show that there exists a formula ψ. As f preserves bisimulations, $\exists y.f(R)(x, y) \wedge Q(y)$ is preserved thus by Theorem 16 is thus equivalent to a modal formula, as required.

Syntactic Characterization To see that q is scoped by an even number of negations in ψ is suffices to note that Q occurs only positively in the right hand side of a formula equivalent to $(\psi[q])^*$. This means q must not occur negatively in an essential way.

Finally using the fact from Example 3.3.3 that new modalities can not be defined by conjoining modalities, there are no two modalities that scope q that are in the scope of the same conjunction. ∎

Corollary 18 *First order definable functions from Kripke models to Kripke models are exactly those that are definable using our three operations $x; y$, $x \cup y$ and $x: B?$.*

We get this from the syntactic considerations of ψ in the proof above.

4 OTHER WORK

4.1 MCCARTHY'S APPROACH

McCarthy (1993) introduced the notion of a context as a formal object. He argued that contexts should objects in a theory. This led him to introduce the modality, $ist(c, \phi)$, which states that a formula ϕ is true in a context c.

McCarthy saw contexts as capturing the unstated features of an utterance. He thought these features could be made more explicit by showing how contexts were related. His proposal was to characterize these relations by writing sentences that relate one context to another, lifting axioms. If ϕ being true in a context c implied that ψ was true in a context c', he would write, $ist(c, \phi) \to ist(c', \psi)$.

While McCarthy was developing the idea of contexts a student of his, Guha (1991) applied the ideas to Cyc (Lenat 1995, Guha and Lenat 1992). In Cyc contexts were used to capture "microtheories", or sets of facts about a particular domain. This compartmentalization of information is useful when data sources from different producers need to be merged, cf. (Farquhar et al. 1995).

Independently Giunchiglia (1993) introduced the notion of a multi-language system. Based on intuitions from proof theory, he attempts to capture how patterns of reasoning in different languages can be integrated. Giunchiglia et al. (1993) address issues of non-omniscient belief, while our contexts are all logically closed. Clearly there is a clear relationship between how patterns of reasoning are related, and theories are related.

4.2 QUOTING APPROACHES

Montague (1963) showed that modal logic cannot be mimicked by a naive quoting machinery if the underlying language is sufficiently powerful. This might be thought to render such approaches impossible, but Attardi and Simi (1995b) have developed a system that avoids these problems. They call their syntactic modalities, *viewpoints*. The key idea of Attardi and Simi is to use reflection principles added to a standard first order logic with names for terms. They show that useful reasoning can be done in consistent theories, while allowing self reference etc., but blocking the paradoxes of Montague.

While quoting approaches are natural to a certain extent, the difficulties that avoiding paradoxes raise justify other approaches, which draw on other intuitions.

4.3 MODAL APPROACHES

Montague (1970) considers introducing modalities as objects in a system extending Montague Grammar, a version of intensional higher order logic. Naturally, his modalities are non-extensional. This led him away from considering structural properties, as his interests lay in the application of these ideas to natural language. Contexts in natural language are intensional, according to many linguists.

Ohlbach, while looking at more efficient ways of reasoning in modal logics (Ohlbach 1988), considered representing modalities in a first order language by reifying the worlds, and explicitly introducing the accessibility relation. This led him to consider *context logics* (Ohlbach 1989). His interest lay mainly in the extra expressive power this gave over standard modalities.

Shoham (1991) considered a class of modal operators each of which characterized a context. He considered many possibilities for uses of contexts, and gives a set of 14 possible axioms for contexts, as a benchmark for comparison. Buvač, Buvač and Mason (1995) have looked at modal logics with these operators and the additional assumption that each operator has complete information about how the other operators behave. That is, their systems assume, $\Box_1\Box_2\phi \vee \Box_1\neg\Box_2\phi$. This axiom does not seem justified, even for the applications they consider. There is no reason why a database should have complete information about the contents of other databases. This limitation has been noticed by (Attardi and Simi 1995a).

(Buvač 1996) considers a simple notion of quantification over contexts, which corresponds to our first completeness theorem, with the assumption $\Box_1\Box_2\phi \vee \Box_1\neg\Box_2\phi$, and a 5 like assumption that multiple modalities may always be reduced to a single modality.

Similar systems to this (Gabbay and Nossum 1997) based on fibering one logic over another (Gabbay 1996) have been proposed. Our focus is on the role quantification over structures should play, an issue that is not the primary focus of fibering. (Akman and Surav 1996) provides a survey of formal approaches to contextual reasoning.

A related area outside AI is the study of programming constructs that are safe for bisimulation (Glabbeek 1990, Glabbeek 1993, van Benthem 1993, Hollenberg 1996). Van Benthem (1993) proves that a first order formula $\phi(x, y)$ is safe for bisimulation if and only if it is expressible by means of atomic transactions, tests of modal formulas, and the operations of relational composition and union. These correspond closely to the functions we propose that make new modalities. The functions we found, with another second order function, *recursion*, are safe for bisimulation in this sense. These constructs were originally developed in the framework of Dynamic Logic, (Pratt 1990). The functions we consider are also suggested by (Catach 1988) as ways of making new modalities.

Bisimulation, and its relation to non-wellfounded set theory are explored in detail in (Barwise and Moss 1996). Non-wellfounded set theory has also been considered as a natural model of *situation theory* (Barwise and Perry 1983) which bears much similarity with context (Akman and Surav 1997, van Benthem 1997), though is more focussed on natural language. Barwise and Moss (1996) show that non-wellfounded set theory, based on bisimulation, is a very natural model of modalities. Thus bisimulation is a unifying theme between the areas of programming constructs, non-wellfounded set theory, situation semantics, and our theory of context.

5 CONCLUSION AND APPLICATIONS

We have presented a sound and complete model theory for a quantificational logic of contexts. We began by considering models which included arbitrary universes of relations. This gave a simple system which allowed quantification over modalities, but which did not have any closure conditions on what modalities should exist.

We then looked at some operations that seemed natural on accessibility relations such as union, intersection and complement. We saw that some operations were not functions from modalities to modalities. The operation intersection can take two modalities and form a relation which does not behave as a modality.

Operations that allow us to take the union of contexts and test a context for satisfaction of given statement do define functions on modalities. These operations are very natural in combining knowledge sources, and considering the knowledge in common of different sources. Taking the union of two modalities results in the intersection of their theories.

Another valid operation on modalities is the relational composition which naturally models the beliefs that one agent has about another. New modalities of this sort are natural when we consider multiple agents.

We then suggested that our universe be closed under these three operators that make new modalities from old modalities. Therefore our universe of modalities includes all modalities definable using these operations. We gave an axiomatization of these operators that we show is sound and complete for universes that include all *first order definable modalities.*

We then showed that these three operators are sufficient to define all first order definable functions on modalities. To prove this theorem we characterized all *first order definable functions on modalities.* We considered when modalities are equivalent. This needed the notion of bisimulation. We then look at what functions on the relations in Kripke models give rise to functions on modalities. We then proved a theorem which characterizes the first order definable functions on relations that give rise to functions on modalities. We showed that the syntactic class of first order definable functions on modalities can be captured by our three operations on modalities.

Acknowledgments

This authors thank Rob van Glabbeek, Fausto Giunchiglia and John McCarthy for many helpful discussions on this and related work. We would also like to thank the attendees of CommonSense '98 and the anonymous referees.

This research was partly funded by DARPA (ONR) grant # N00014-94-0775 under ARPI and by the Air Force Office of Scientific Research Grant # F49620-97-1-0207 under the DARPA HPKB project and New World Vistas.

References

Akman, V., and M. Surav. 1996. Steps toward formalizing context. *AI Magazine* 17(3):55–72.

Akman, V., and M. Surav. 1997. The use of situation theory in context modeling. *Computational Intelligence* 13(3):427–438.

Attardi, G., and M. Simi. 1995a. Beppe had a dream. In *AAAI 1995 Fall Symposium on Formalizing Context.* AAAI.

Attardi, G., and M. Simi. 1995b. A formalization of viewpoints. *Fundamenta Informaticae* 23(3).

Barwise, J., and L. Moss. 1996. *Vicious Circles.* CSLI.

Barwise, J., and J. Perry. 1983. *Situations and Attitudes.* Cambridge: MIT Press.

Benthem, J. v. 1983. *Modal Logic and Classical Logic.* Naples: Bibliopolis.

Buvač, S. 1996. Quantificational logic of context. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence.*

Buvač, S., V. Buvač, and I. A. Mason. 1995. Metamathematics of contexts. *Fundamenta Informaticae* 23(3).

Catach, L. 1988. Normal Multimodal Logics. In *Proc. National Conference on Artificial Intelligence (AAAI '88)*, 491–495, Saint Paul, MI. AAAI.

Farquhar, A., A. Dappert, R. Fikes, and W. Pratt. 1995. Integrating information sources using context logic. Technical Report KSL-95-12, Knowledge Systems Laboratory, Stanford University. Also appears in the 1995 AAAI Spring Symposium Series on Information Gathering from Distributed, Heterogeneous Environments.

Gabbay, D. 1996. Fibred semantics and the weaving of logics: Part I: Modal and intuitionistic logics. *Journal of Symbolic Logic* 61(4):1057–1120.

Gabbay, D., and R. Nossum. 1997. Structured contexts with fibred semantics. In *Proceedings of the International and Interdisciplinary Conference on Context.*

Giunchiglia, F. 1993. Contextual reasoning. *Epistemologia, special issue on I Linguaggi e le Macchine* XVI:345–364.

Giunchiglia, F., L. Serafini, E. Giunchiglia, and M. Frixione. 1993. Non-omniscient belief as context-based reasoning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence.*

Glabbeek, R. v. 1990. The linear time – branching time spectrum (extended abstract). In J. Baeten and J. Klop (Eds.), Proceedings *CONCUR '90, Theories of Concurrency: Unification and Extension,* Amsterdam, August 1990, Vol. 458, 278–297.

Glabbeek, R. v. 1993. The linear time – branching time spectrum II; the semantics of sequential systems with silent moves (extended abstract). In E. Best (Ed.), Proceedings *CONCUR'93*, 4th International Conference on Concurrency Theory, Hildesheim, Germany, August 1993, Vol. 715, 66–81.

Guha, R. V. 1991. *Contexts: A Formalization and Some Applications.* PhD thesis, Stanford University.

Guha, R., and D. Lenat. 1992. Language, representation and contexts. *Journal of Information Processing* 15(3):340–349.

Halpern, J. 1995. Reasoning About Knowledge: A Survery. In D. Gabbay and J. Hogger, C. Robinson (Eds.), *Handbook of Logic in Artificial Intelligence and Logic Programming*, Vol. IV, 1–34. Clarendon Press, Oxford.

Hollenberg, M. 1996. Safety for bisimulation in monadic second-order logic. Logic Group Preprint Series 170, Dept. of Philosophy, Utrecht University, November.

Lenat, D. B. 1995. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the Association for Computing Machinery* 38 (11).

McCarthy, J. 1993. Notes on formalizing context. In *Proc. Thirteenth International Joint Conference on Artificial Intelligence (IJCAI '93).*

Montague, R. 1963. Syntactical treatments of modality, with corollaries on reflection principles and finite axiomatizations. *Acta Philosophica Fennica* 16:153–167.

Montague, R. 1970. Universal grammar. *Theoria* 36:373–398.

Ohlbach, H. J. 1988. *A Resolution Calculus for Modal Logics.* PhD thesis, Kaiserslautern.

Ohlbach, H. J. 1989. Context logic. Technical report, Kaiserslautern.

Pratt, V. 1990. Dynamic algebras as a well-behaved fragment of relation algebras. In *Algebraic Logic and Universal Algebra in Computer Science, LNCS 425*, 77–110, Ames, Iowa, June 1988. Springer-Verlag.

Shoham, Y. 1991. Varieties of context. In V. Lifschitz (Ed.), *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy.* Academic Press.

van Benthem, J. 1993. Programming operations that are safe for bisimulation. Technical Report 93-179, CSLI, Center for the Study of Language and Information, Stanford University. To appear in *Studai Logica.*

van Benthem, J. 1997. Changing contexts and shifting assertions. In *Computing Natural Language.* CSLI Publications. Distributed by Cambridge University Press.

Local Models Semantics, or Contextual Reasoning = Locality + Compatibility

Fausto Giunchiglia
ITC-IRST
Povo, 38100 Trento, Italy
fausto@irst.itc.it

Chiara Ghidini
DISA - University of Trento,
Via Inama 5, 38100 Trento, Italy
ghidini@cs.unitn.it

Abstract

In this paper we present a new semantics, called *Local Models Semantics*, and use it to provide a foundation to reasoning with contexts. This semantics captures and makes precise the two main intuitions underlying contextual reasoning: *(i)* reasoning is mainly *local* and uses only part of what is potentially available (e.g. what is known, the available inference procedures), this part is what we call *context* (of reasoning); however *(ii)* the context may change and there is *compatibility* among the reasoning performed in different contexts. We validate our semantics by formalizing two important forms of contextual reasoning: reasoning with viewpoints and reasoning about belief.

1 INTRODUCTION

Contexts are an important topic in many research areas, for example in the semantics of natural language, in computational linguistics, and in cognitive science. Lately, contexts have been independently proposed in (Giunchiglia, 1993) and (McCarthy, 1987) as an important means for formalizing (certain forms of) reasoning. In (Giunchiglia, 1993) contexts are seen as a tool for formalizing the locality of reasoning, while in (McCarthy, 1987) contexts are seen as a way for solving the problem of generality. Coherently with these two proposals, contexts have been used in various applications, e.g. in the integration of heterogeneous knowledge and data bases (Farquhar, Dappert, Fikes and Pratt, 1995; Mylopoulos and Motschnig-Pitrik, 1995), in the formalization of reasoning about beliefs (Giunchiglia, 1993; Giunchiglia and Serafini, 1994), and in the formalization of reasoning with viewpoints (Attardi and Simi, 1995).

Despite the pletora of different approaches, formalizations, and applications, two are the main intuitions underlying the use of context in reasoning (and also in the other research areas). We state these two intuitions as the following two principles:

Principle 1 (of locality): reasoning uses only part of what is potentially available (e.g. what is known, the available inference procedures). The part being used while reasoning is what we call *context* (of reasoning);

Principle 2 (of compatibility): the context may change. However, there is *compatibility* among the reasoning performed in different contexts.

The goal of this paper is to describe and motivate a new semantics, called *Local Models Semantics (LMS)*, which formalizes the two principles listed above, and that we propose as a foundation of contextual reasoning. In Section 2 we give two motivating examples. In Section 3 we present LMS and discuss how it formalizes the principles of locality and compatibility, plus other intuitions. In Section 4 we formalize the two examples introduced in Section 2. Finally, in Section 5 we compare LMS with other frameworks proposed in the past. We end with some concluding remarks (Section 6).

2 TWO MOTIVATING EXAMPLES

The first example is about reasoning with viewpoints, the second is about reasoning about belief. These examples highlight how the semantics and methodology developed in this paper can be applied, suitably generalized, to the modelling of two important problems: the federation of heterogeneous data or knowledge bases (first example) and multiagent systems (second example).

2.1 VIEWPOINTS

Consider the scenario of Figure 1. There are two observers, $Mr.1$ and $Mr.2$, each having a partial view of a box. The box consists of six sectors, each sector possibly containing a ball. There cannot be balls hidden from the view of an observer. The box is "magic" and observers cannot distinguish the depth inside it. We have two contexts, each context for what an observer sees and the consequences that it is able to draw from it.

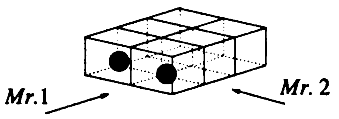


Figure 1: The Magic Box.

Locality: Both $Mr.1$ and $Mr.2$ have the notions of a ball being on the right or on the left. However these two notions are different and we may have a ball which is on the right for $Mr.1$ and not on the right for $Mr.2$. Furthermore $Mr.2$ has the notion of "a ball being in the center of the box" which is meaningless for $Mr.1$.

Compatibility: $Mr.1$ and $Mr.2$'s views are viewpoints of the same box and, therefore, there must be a correspondence between them. Figure 2 gives all the views and possible combinations of compatible views of $Mr.1$ and $Mr.2$. We can describe this situation by listing all the possible compatible pairs (exactly as they are represented in Figure 2), or we can describe

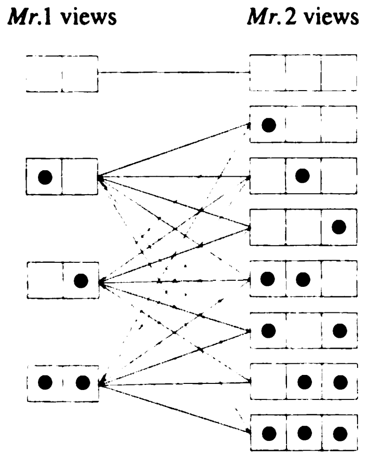


Figure 2: Compatible Views of $Mr.1$ and $Mr.2$.

it more synthetically, and possibly also less precisely. For instance we can have a description like: "if $Mr.1$ sees at least a ball then $Mr.2$ sees at least a ball".

Notice that the most straightforward formalization would be a direct axiomatization of the box as a two-dimensional grid. $Mr.1$ and $Mr.2$'s views could then be easily constructed by projecting the grid in two one-dimensional views. Locality and compatibility would be guaranteed by construction. However this approach is based on the hypothesis that we have a complete description of the world (the box in this case), and that we can use it to build views of the world itself. This is not always the case. Quite often we have only partial views and we can only reconstruct a partial compatibility among them. An important application domain where this may or may not be the case is the development and integration of data or knowledge bases. Thus in a relational, possibly distributed, data base there is (what is assumed to be) a complete description of the world, and views are built by filtering out and appropriately merging together part of the available information. On the other hand, a federation of heterogeneous data or knowledge bases, possibly developed independently, can be seen as a set of views of an ideal data base which is often impossible or very complex to reconstruct completely. The work in (Serafini and Ghidini, 1997) starts from this observation, further develops the semantics defined in this paper, and gives foundations to the various kinds of federation described in, e.g., (Farquhar et al., 1995; Mylopoulos and Motschnig-Pitrik, 1995).

2.2 REASONING ABOUT BELIEF

Consider now the situation of two agents, a and b with a having beliefs about the beliefs of b (see Figure 3). We have again two contexts, one for what a believes and one for what b believes.

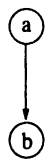


Figure 3: a has Beliefs About the Beliefs of b.

Locality: a and b have different beliefs about the world. Furthermore, a has the notion of "b believing something" which b does not have.

Compatibility: The level of compatibility depends on the situation. We may, for instance, have that a's beliefs about b's beliefs are correct, in which case if a believes that b believes ϕ then b believes ϕ. Another situation is when a's beliefs about b's beliefs are complete, in which case if b believes ϕ then a believes that

b believes ϕ. A taxonomy of the possible forms of belief about belief is introduced in (Giunchiglia, Serafini, Giunchiglia and Frixione, 1993).

The presentation of a multiagent system as a set of contexts, each context representing an agent, was first advocated in (Giunchiglia et al., 1993). In this approach the reasoning capabilities of an agent, its interactions with the other agents, and its beliefs about the beliefs of the other agents can be directly modelled as properties of the context representing the agent. A specification given in this way has properties of modularity and incrementality which are missing in other frameworks. The specifications developed in (Giunchiglia et al., 1993) were axiomatic and set-theoretic. This paper highlights how this approach can be given a semantic foundation which maintains modularity and incrementality.

3 LOCAL MODELS SEMANTICS

For the sake of simplicity, let us consider the situation with two contexts (the generalization to any enumerable set of contexts does not present conceptual difficulties). Let us restrict ourselves to (classes of) first order models and let us suppose that L_1, L_2 are the two first order languages used to describe what is true in the two contexts. Let $\overline{M}_i$ be the class of all the models (interpretations) of L_i. Let $\mathbf{C}$,

$$\mathbf{C} \subseteq 2^{\overline{M}_1} \times 2^{\overline{M}_2},$$

be a *compatibility relation* (for $\{L_1, L_2\}$). Let us call $m \in \overline{M}_i$ a *local model* (of L_i) and $\mathbf{c} \in \mathbf{C}$ a *(compatibility) pair* (for $\{L_1, L_2\}$). Notationally, if $\mathbf{c}$ is a pair, we write $\mathbf{c}_1$ and $\mathbf{c}_2$ to mean its first and second element, respectively; and $\mathbf{c}_i$ to mean its i-th element, with i possibly equal to 1 or 2.

A *model* is a compatibility relation which contains at least a pair and does not contain the pair of two empty sets.

Definition 3.1 (Model) *A* model *(for $\{L_1, L_2\}$) is a compatibility relation* $\mathbf{C}$ *such that:*

1. $\mathbf{C} \neq \emptyset$;
2. $\langle \emptyset, \emptyset \rangle \notin \mathbf{C}$.

In the following we write $\mathbf{C}$ to mean either a compatibility relation or a model. The context always makes clear what we mean. Figure 4 gives a graphical representation of the construction we perform. We start from L_1, L_2. Then, we associate each of them with a set $M_i \subseteq \overline{M}_i$ of local models. Usually $M_i \subset \overline{M}_i$. For instance, in the magic box scenario, if there is only one ball in the box, then all the local models (for both $Mr.1$ and $Mr.2$) which allow zero, two or three balls must be discarded. Finally, we pair local models inside compatibility pairs. The resulting compatibility relation is our model. For instance, we can construct a local model for each view in Figure 2 and then build a compatibility relation which contains the pairs constructed as follows: as first element take a singleton set containing a model formalizing a view of $Mr.1$, as second element take a singleton set containing a model formalizing a view compatible with the chosen view of $Mr.1$. Notice that linking local models inside a compatibility relation may force us to throw away some of them. For instance, considering the local models for $Mr.1$ which allow for only one ball forces us to throw away all the pairs and corresponding local models for $Mr.2$ which allow for zero balls.

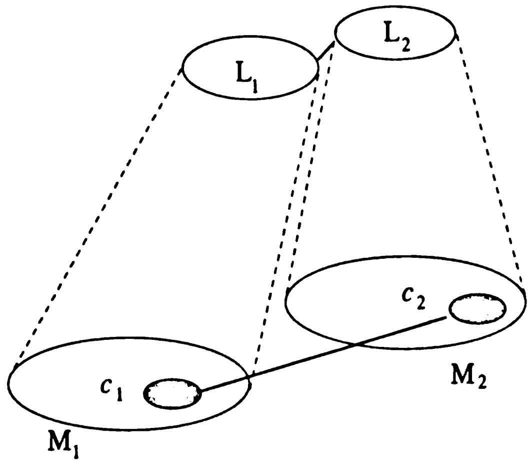


Figure 4: Model for $\{L_1, L_2\}$.

A *context* C_i, with $i = 1, 2$, (in a model $\mathbf{C}$) is (formally defined as) the set of local models $m \in \overline{M}_i$ allowed by $\mathbf{C}$, in symbols

$$C_i = \{m \ : \ m \in \mathbf{c}_i \text{ with } \mathbf{c} \in \mathbf{C}\}.$$

Notice that defining a context as a set of models (instead of, e.g., a single model) allows us to formalize it as a partial object, as explicitly required in, e.g., (Giunchiglia, 1993; McCarthy, 1987). This is a key difference with possible worlds, which are complete objects (in the sense that a formula is either true or false in a world). The advantages of having contexts as partial objects are easily seen. Consider for instance the example of Section 2.2. In this case we may impose that a is uncommitted to whether b believes ϕ simply by having ϕ true in some local models of the context for a and false in others. A similar observation applies to compatibility pairs. For instance, in the example of Section 2.1, we can impose a compatibility constraint which says that if $Mr.1$ sees at least a ball then $Mr.2$ sees at least a ball. In this case we have in the same compatibility pair local models for $Mr.1$ with a ball on the left and models where this is not the case, and analogously for $Mr.2$.

We can now say what it means for a model to *satisfy* a formula of a language L_i. Let $\models_{cl}$ be the (classical) satisfiability relation between local models and formulas of L_i. Let us call $\models_{cl}$ *local satisfiability*. Notationally, 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 or, also, a labelled L_i-formula. This notation and terminology allows us to keep track of the context we are talking about. Then we have the following:

Definition 3.2 (Satisfiability) *Let* $\mathbf{C}$ *be a model and* $i: \phi$ *a formula. Then,* $\mathbf{C}$ satisfies $i: \phi$, *in symbols* $\mathbf{C} \models i: \phi$ *iff, for all* $\mathbf{c} \in \mathbf{C}$, $\mathbf{c}_i \models i: \phi$, *where* $\mathbf{c}_i \models i{:}\phi$ *iff, for all* $m \in \mathbf{c}_i$, $m \models_{cl} \phi$.

Intuitively: a formula of the context C_i is satisfied by a model $\mathbf{C}$ if all the local models in C_i satisfy it. A model $\mathbf{C}$ satisfies a set of formulas Γ, in symbols $\mathbf{C} \models \Gamma$, iff $\mathbf{C}$ satisfies every formula $i{:}\phi \in \Gamma$.

The notions of model, context and satisfiability given above formalize the principles of locality and compatibility in the following sense:

Locality: Everything is local. First of all, the language is local: not only do we have a language for each context, but, also, there is no notion of a not labelled L_i-formula ϕ being satisfiable. We always talk of satisfiability of formulas in context, i.e., of labelled L_i-formulas. Second, the notion of satisfiability is local: the satisfiability of a (labelled) formula is given in terms of the local satisfiability of the formula with respect to its context. Third, the structures we consider to test local satisfiability are local: contexts have their own, generally different, domains of interpretation, sets of relations, and sets of functions.

Compatibility: Via compatibility, contexts mutually influence themselves. Compatibility has the structural effect of changing the set of local models defining each context. It forces local models to agree up to a certain extent. On one extreme the two contexts have two independent views of the world. In this case we have a compatibility relation which is the empty set and there is no compatibility between what holds in the distinct sets of local models. On the other extreme the two contexts describe the same world. In this case all the languages are the same, and for every local model in a context there is a corresponding compatible identical local model in the other context. Furthermore, the compatibility relation is the identity mapping. In this case all the contexts are a replication of the same context and we are in the classical situation of one language and one notion of satisfiability and truth.

The notion of *validity* is the obvious one. A formula $i{:}\phi$ is *valid*, in symbols $\models i{:}\phi$, if and only if all models satisfy $i{:}\phi$.

What is more interesting is the notion of *logical consequence* which must take into account the fact that assumptions and conclusion may belong to distinct languages. Let Γ_j be a set of labelled L_j-formulas, and $\mathbf{c}_j$ be an element of a compatibility pair. Notationally, we write $\mathbf{c}_j \models \Gamma_j$ to mean that $\mathbf{c}_j \models j: \gamma$ for every $j{:}\gamma \in \Gamma_j$.

Definition 3.3 (Logical Consequence) *A formula* $i{:}\phi$ *is a* logical consequence *of a set of formulas* $\Gamma = \cup_{j=1,2}\Gamma_j$, *in symbols* $\Gamma \models i{:}\phi$, *if and only if, for all models* $\mathbf{C}$, *for all* $\mathbf{c} \in \mathbf{C}$, *if for all* $j \in \{1,2\}$, $j \neq i$, $\mathbf{c}_j \models \Gamma_j$, *then for all* $m \in \mathbf{c}_i$, *if* $m \models_{cl} \Gamma_i$ *then* $m \models_{cl} \phi$

Intuitively: take a formula $i: \phi$. Take a set of assumptions Γ and, among them, isolate the set of assumptions Γ_j with $j \neq i$. Take all the pairs whose local models in C_j satisfy Γ_j (and throw away all the others). Consider now the local models in C_i of the remaining pairs. $\Gamma \models i: \phi$ if in these remaining local models all the local models which satisfy Γ_i locally satisfy ϕ. Essentially, the idea is that the formulas in Γ_j prune away compatibility pairs, while the formulas in Γ_i prune away local models in C_i.

Notice that the notion of logical consequence defined above reduces to the classical notion of logical consequence in the case of a single language.

4 THE TWO EXAMPLES – FORMALIZED

4.1 VIEWPOINTS

Let us start by defining L_1 and L_2. Let $P_1 = \{r, l\}$ and $P_2 = \{r, c, l\}$ be two sets of propositional constants (intuitively, r, c, l stand for ball on the right, in the center and on the left, respectively). L_1 and L_2 are defined as the smallest set containing P_1 and closed under the propositional connectives, and the smallest set containing P_2 and closed under the propositional connectives, respectively. Notice that L_1 (L_2) can describe all the views of $Mr.1$ ($Mr.2$) but that, for instance, L_1 cannot describe the situation with a ball in the center (that is, a possible view of $Mr.2$).

L_1 and L_2 have the usual Tarskian propositional semantics. The local models of L_1 are (univocally defined by the following sets of formulas):

$$\begin{array}{ll} m_1 = \{\neg l, \neg r\} & m_2 = \{l, \neg r\} \\ m_3 = \{\neg l, r\} & m_4 = \{l, r\}. \end{array}$$

The local models of L_2 are:

$$\begin{array}{ll} m_1 = \{\neg l, \neg c, \neg r\} & m_2 = \{l, \neg c, \neg r\} \\ m_3 = \{\neg l, c, \neg r\} & m_4 = \{l, c, \neg r\} \\ m_5 = \{\neg l, \neg c, r\} & m_6 = \{l, \neg c, r\} \\ m_7 = \{\neg l, c, r\} & m_8 = \{l, c, r\}. \end{array}$$

Let us construct a model for the magic box by imposing the following compatibility constraints:

1. if $Mr.1$ sees at least a ball then $Mr.2$ sees at least a ball;
2. if $Mr.2$ sees at least a ball then $Mr.1$ sees at least a ball;
3. $Mr.1$ and $Mr.2$ both see a box, or neither does.

The key notion in defining a compatibility relation is the identification of the set of L_i-formulas which are satisfied in all the local models in $\mathbf{c}_i$. Let us write $\Theta(\mathbf{c}_i)$ to mean such a set. $\Theta(\mathbf{c}_i)$, with $i = 1, 2$, characterizes the kind of compatibility imposed by a pair.

Definition 4.1 (A model for the magic box) *A model* $\mathbf{C}$ *for the magic box is a compatibility relation such that, for all* $\mathbf{c} \in \mathbf{C}$

$$\textit{if } l \vee r \in \Theta(\mathbf{c}_1) \textit{ then } l \vee c \vee r \in \Theta(\mathbf{c}_2); \quad (1)$$

$$\textit{if } l \vee c \vee r \in \Theta(\mathbf{c}_2) \textit{ then } l \vee r \in \Theta(\mathbf{c}_1); \quad (2)$$

$$\mathbf{c}_1 \neq \emptyset \textit{ iff } \mathbf{c}_2 \neq \emptyset. \quad (3)$$

Equation (1) models constraint 1. In fact, if $Mr.1$ sees a ball then this ball can only be on the left or on the right. In either case $l \vee r \in \Theta(\mathbf{c}_1)$, for all $\mathbf{c}$. Furthermore, in this case $Mr.2$ sees a ball in one of the three possible positions, and, therefore, $l \vee c \vee r \in \Theta(\mathbf{c}_2)$. A similar explanation can be given for Equation (2). Constraint 3 is more interesting. Equation (3) says that $Mr.1$ and $Mr.2$ have a model of the box only if they see it, and that this must hold for both of them, or for neither.

Example 4.1 Many models satisfying Definition 4.1 can be constructed. Let us consider the model(s) whose pairs only contain at most singleton sets. It is easy to see that in these models, if $Mr.1$ sees no balls then $Mr.2$ sees no balls, formally, $1{:}\neg l \wedge \neg r \models 2{:}\neg l \wedge \neg c \wedge \neg r$. To prove this, let us consider all the pairs $\langle \mathbf{c}_1, \mathbf{c}_2 \rangle$ such that $\mathbf{c}_1$ satisfies $1{:}\neg l \wedge \neg r$. From Definition 3.3 we must show that every $\mathbf{c}_2$ satisfies $2{:}\neg l \wedge \neg c \wedge \neg r$. Suppose that there exists a $\mathbf{c}_2$ such that $\mathbf{c}_2$ does not satisfy $2{:}\neg l \wedge \neg c \wedge \neg r$. $\mathbf{c}_2$ contains (at most) a propositional local model. Therefore, $\mathbf{c}_2$ satisfies $2{:}l \vee c \vee r$ and $l \vee c \vee r \in \Theta(\mathbf{c}_2)$. From Equation (2) we obtain $l \vee r \in \Theta(\mathbf{c}_1)$. $\mathbf{c}_1$ is a propositional model and it cannot satisfy both $1{:}\neg l \wedge \neg r$ and $1{:}l \vee r$. Thus $\mathbf{c}_1 = \emptyset$. By Equation (3), $\mathbf{c}_2 = \emptyset$ as well. This cannot be possible for Definition 3.1. Therefore the hypothesis that $\mathbf{c}_2$ does not satisfy $2 : \neg l \wedge \neg c \wedge \neg r$ must be false. Thus $1{:}\neg l \wedge \neg r \models 2{:}\neg l \wedge \neg c \wedge \neg r$. In a similar way, we can also prove the vice versa, that is, $2{:}\neg l \wedge \neg c \wedge \neg r \models 1{:}\neg l \wedge \neg r$. Furthermore, it can be shown that, in Definition 4.1, Equation (3) can be substituted with the following equation

$$\neg l \wedge \neg r \in \Theta(\mathbf{c}_1) \text{ iff } \neg l \wedge \neg c \wedge \neg r \in \Theta(\mathbf{c}_2) \quad (4)$$

which states exactly the two facts proved above. It is easy to notice that Equations (1), (2), (4) directly capture, as pairs with only singleton sets, all the compatible views in Figure 2 (Equation (4) capturing the one at the top). As a consequence the model which contains *all* the possible pairs containing only singleton sets is the model which directly formalizes the compatible views in Figure 2. The pairs in this model are as follows:

$$\left\{ \begin{array}{c} \langle \{\{\neg l, \neg r\}\}, \{\{\neg l, \neg c, \neg r\}\} \rangle, \\ \langle \{\{l, \neg r\}\}, \{\{l, \neg c, \neg r\}\} \rangle, \\ \langle \{\{l, \neg r\}\}, \{\{\neg l, c, \neg r\}\} \rangle, \\ \cdots \\ \langle \{\{l, r\}\}, \{\{l, c, r\}\} \rangle \end{array} \right\}$$

All the models whose pairs contain only singleton sets are subsets of this model.

4.2 REASONING ABOUT BELIEF

Let us start by defining L_a and L_b. L_a and L_b are two first order languages. L_a contains a predicate B and a term "ϕ" for every formula $\phi \in L_b$. $B(\text{“}\phi\text{”})$ intuitively stands for "b believes ϕ". To make things more concrete, let us suppose that b has beliefs about a young student, named Paul. To this extent, let us add an individual constant p (which stands for Paul) and two unary predicates S and Y (intuitively meaning "to be student" and "to be young", respectively) to L_b.

L_a and L_b have the usual Tarskian semantics. The local models of L_a are pairs $\langle \mathbf{dom}_a, \mathcal{I}_a \rangle$, where: $\mathbf{dom}_a$ is the domain of interpretation of a, and $\mathcal{I}_a$ maps every term "ϕ" to a constant $d_\phi \in \mathbf{dom}_a$, and the predicate

B to a relation $R_B \subseteq \mathbf{dom}_a$. The local models of L_b are pairs $\langle \mathbf{dom}_b, \mathcal{I}_b \rangle$ where: $\mathbf{dom}_b$ is the domain of interpretation of b, and $\mathcal{I}_b$ is an interpretation function mapping the individual constant p to a constant $d_p \in \mathbf{dom}_b$ and the predicates S and Y to the relations $R_S \subseteq \mathbf{dom}_b$ and $R_Y \subseteq \mathbf{dom}_b$, respectively.

The key part in the construction of models of reasoning about the belief is the definition of the relation existing between the beliefs of b and the beliefs of a of the form $B(\text{"}\phi\text{"})$. To this extent let us introduce the following notation: Let Γ be a set of L_b-formulas. We write $B(\text{"}\Gamma\text{"})$ to mean the set of L_a-formulas $B(\text{"}\phi\text{"})$ such that ϕ belongs to Γ. Let Γ be a set of L_a-formulas. We write $B^{-1}(\text{"}\Gamma\text{"})$ to mean the set of L_b-formulas ϕ such that $B(\text{"}\phi\text{"})$ belongs to Γ.

Let us define a model with a correct and complete with respect to b's beliefs. In this case "b believes ϕ" belongs to a's beliefs iff ϕ belongs to b's beliefs.

Definition 4.2 (A Model for reasoning about belief) *A model* $\mathbf{C}$ *for reasoning about belief is a compatibility relation such that, for all* $\mathbf{c} \in \mathbf{C}$

$$B^{-1}(\text{"}\Theta(\mathbf{c}_a)\text{"}) \subseteq \Theta(\mathbf{c}_b) \qquad (5)$$

$$B(\text{"}\Theta(\mathbf{c}_b)\text{"}) \subseteq \Theta(\mathbf{c}_a) \qquad (6)$$

(5) models the soundness of a; (6) models its completeness. Let us explain (6) ((5) can be explained analogously). Take $\Theta(\mathbf{c}_b)$, that is all the L_b-formulas which are true in $\mathbf{c}_b$. $\Theta(\mathbf{c}_b)$ is the set of the beliefs of b, at least for what concerns $\mathbf{c}$ (taking a belief to be a formula which is true in all local models). Take now the set of a's beliefs about the actual beliefs of b, that is $B(\text{"}\Theta(\mathbf{c}_b)\text{"})$ (again restricted to $\mathbf{c}$). Then for a to be complete (but not necessarily correct) we must have that $B(\text{"}\Theta(\mathbf{c}_b)\text{"})$ is a subset of its beliefs, in symbols $B(\text{"}\Theta(\mathbf{c}_b)\text{"}) \subseteq \Theta(\mathbf{c}_a)$.

Example 4.2 Suppose that a believes that b believes that Paul is a student and suppose that b believes that every student is young. Does a believe that b believes that Paul is young? Formally, we want to verify whether $a\colon B(\text{"}Y(p)\text{"})$ is a consequence of the two axioms $a\colon B(\text{"}S(p)\text{"})$ and $b\colon \forall x.S(x) \supset Y(x)$. All models $\mathbf{C}$ (and all $\mathbf{c}_i$) satisfy $a\colon B(\text{"}S(p)\text{"})$ (being this formula an axiom). From the definition of satisfiability $B(\text{"}S(p)\text{"}) \in \Theta(\mathbf{c}_a)$ for every $\mathbf{c}_a$. From Equation (5), $S(p) \in \Theta(\mathbf{c}_b)$ for every $\mathbf{c}_b$ and thus $\mathbf{C}$ satisfies $b\colon S(p)$. All models $\mathbf{C}$ satisfy $b\colon \forall x.S(x) \supset Y(x)$. From the definition of satisfiability, every local model of L_b in $\mathbf{C}$ satisfies $\forall x.S(x) \supset Y(x)$. Local models are classical models, thus they satisfy $S(p) \supset Y(p)$ and, therefore, $Y(p)$. It follows that $\mathbf{C}$ satisfies $b\colon Y(p)$, i.e. $b\colon Y(p) \in \Theta(\mathbf{c}_b)$. By applying Equation (6) we obtain $a\colon B(\text{"}Y(p)\text{"}) \in \Theta(\mathbf{c}_a)$, i.e. the model satisfies $a\colon B(\text{"}Y(p)\text{"})$. This completes the proof.

It is important to notice that B is a predicate. Its extension are the formulas it holds on. It is natural to ask what is the relation between the semantics proposed here and other approaches, e.g., possible worlds semantics, where belief is formalized as a modal operator. Section 5 provides an informal but detailed comparison between Local Models Semantics and possible worlds semantics in general. Limited to belief, we start by observing that the extension of B and the beliefs of b are defined independently, and that Equations (5) and (6) only provide a posteriori compatibility constraints. This feature is (in part, see also Section 5) the semantic counterpart of the modularity and incrementality hinted in Section 2 and discussed at the proof-theoretical level in (Giunchiglia et al., 1993). It allows us to define the kind (and quality) of a's beliefs about beliefs by playing with equations like Equations (5) and (6). For instance, it can be shown that LMS scales up very naturally from very strong forms of non-omniscient belief up to omniscient belief, and that it allows for a very elegant treatment of bounded beliefs. Technically, it is possible to prove completeness and correctness results with the proof calculi for belief (modeled as contextual reasoning) described in (Giunchiglia and Serafini, 1994; Giunchiglia et al., 1993). (These proofs are not given here for lack of space.). The proof calculi presented in these two papers (and others) can be proved equivalent with the various normal and non normal modal logics, e.g., those described in (Fagin and Halpern, 1988). All these results sound as follows. Take a modal logic X_M and the corresponding logic for contextual reasoning X_C. Then

$$\vdash_{X_M} \phi \text{ iff } \vdash_{X_C} a\colon \phi.$$

In other words, a (or, in the case of a hierarchy of agents, the agent at the top of the hierarchy) proves the same set of theorems as the modal logic under consideration. The logic for reasoning about belief defined in this section allows for no nesting of B, and makes B weaker than the modal operator of modal K.

5 OTHER FRAMEWORKS – A COMPARISON

The obvious, most studied, framework to start from is possible worlds semantics. Both LMS and possible worlds semantics allow for multiple objects (models or worlds) and have a notion of local satisfiability (to a local model or to a possible world). However there are also some important differences. First, in possible worlds there is a unique language which describes what is true in all the worlds and there is no notion of truth of a labelled formula. This is the case also for the extensions of possible worlds semantics aimed at formalizing local reasoning (see, e.g., (Fagin and Halpern, 1988)), where localization is achieved by adding a new modal operator to the language. Second, worlds are not (Tarskian) models, the key difference being that possible worlds allow for the use of modal operators. The satisfiability of a formula containing a modal operator is defined in terms of the accessibility relation, which must therefore be given before defining satisfiability in a world. The notion of satisfiability in a world is a function of the model of which the world is part. This is not the case for LMS where each local model has its own notion of satisfiability, and where the model and its structure influence only the set of local models under consideration. The hypothesis of using a single unique global language and of being able to describe a priori the structure of the model under consideration is very useful and works in many situations. It does not seem to work in those cases where there is no global scheme describing the system, e.g., the federation of heterogeneous data or knowledge bases or multiagent systems.

In the last few years various semantics for contextual reasoning have been proposed. Most of them are based on possible worlds semantics. As far as we know, the first attempt is described in (Giunchiglia, Serafini and Simpson, 1992). In this work there is a notion of labelled formula and of (local) satisfiability to a set of (local) possible worlds. This semantics works well for contextual logics equivalent to modal K or stronger. Its main limitation is that it is not clear how to extend it to other logics, e.g., non normal modal logics or logics for reasoning with viewpoints.

Guha, in his PhD thesis (Guha, 1991) informally describes a semantics for reasoning with context. Understanding Guha's informal definitions is a non-trivial task. Some of the main ideas seem the following. There is a single global language from which it is possible to extract the (local) languages of all the contexts. There seems to be a notion of satisfaction of labelled formulas, and a notion of labelled formulas being meaningless in a context. There is distinguished symbol `ist`, whose intuitive meaning is "is true", which seems treated as a modal operator. Guha's semantics has been partially formalized in the work by Buvac and his co-authors (see for instance (Buvac and Mason, 1993)). Buvac's semantics seems to have the same features and defects as the semantics in (Giunchiglia et al., 1992), with the further complication that, starting from a single language, there is a lot of work to do in order to achieve locality. In particular the formulas of the global language which are meaningless in a context must be treated as such (this is done using Bochvar three valued logic).

6 CONCLUSION

In this paper we have presented a new semantics, called Local Models Semantics, and proposed it as a foundation for reasoning with context. LMS formalizes the two general principles underlying contextual reasoning, namely the principle of locality and the principle of compatibility. In this paper we have shown how LMS can be used to model two important forms of contextual reasoning, namely reasoning with viewpoints and reasoning about belief.

Acknowledgements

This work has benefited from many discussions with Luciano Serafini and Paolo Bouquet.

References

Attardi, G. and Simi, M. (1995), 'A formalisation of viewpoints', *Fundamenta Informaticae* **23**(2–4), 149–174.

Buvac, S. and Mason, I. A. (1993), 'Propositional logic of context', *Proc. of the 11th National Conference on Artificial Intelligence* .

Fagin, R. and Halpern, J. (1988), 'Belief, awareness, and limited reasoning', *Artificial Intelligence* **34**, 39–76.

Farquhar, A., Dappert, A., Fikes, R. and Pratt, W. (1995), Integrating Information Sources Using Context Logic, *in* 'Proceedings of AAAI Spring Symposium on Information Gathering from Distributed Heterogeneous Environments'.

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, F. and Serafini, L. (1994), 'Multilanguage hierarchical logics (or: how we can do without modal logics)', *Artificial Intelligence* **65**, 29–70. Also IRST-Technical Report 9110-07, IRST, Trento, Italy.

Giunchiglia, F., Serafini, L., Giunchiglia, E. and Frixione, M. (1993), Non-Omniscient Belief as Context-Based Reasoning, *in* 'Proc. of the 13th International Joint Conference on Artificial Intelligence', Chambery, France, pp. 548–554. Also IRST-Technical Report 9206-03, IRST, Trento, Italy.

Giunchiglia, F., Serafini, L. and Simpson, A. (1992), Hierarchical meta-logics: intuitions, proof theory and semantics, *in* 'Proc. of META-92, Workshop on Metaprogramming in Logic', number 649 *in* 'LNCS', Springer Verlag, Uppsala, Sweden, pp. 235–249. Also IRST-Technical Report 9101-05, IRST, Trento, Italy.

Guha, R. (1991), Contexts: a Formalization and some Applications, PhD thesis, Department of Computer Science, Stanford University. Also report No. STAN-CS-91-1399-Thesis.

McCarthy, J. (1987), 'Generality in Artificial Intelligence', *Communications of ACM* **30**(12), 1030–1035. Also in V. Lifschitz (ed.), *Formalizing common sense: papers by John McCarthy*, Ablex Publ., 1990, pp. 226–236.

Mylopoulos, J. and Motschnig-Pitrik, R. (1995), Partitioning Information Bases with Contexts, *in* 'Third International Conference on Cooperative Information Systems', Vienna.

Serafini, L. and Ghidini, C. (1997), Context Based Semantics for Federated Databases, *in* 'Proceedings of the 1st International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT-97)', Rio de Jeneiro, Brazil, pp. 33–45. Also IRST-Technical Report 9609-02, IRST, Trento, Italy.

Reasoning about Actions II

Concurrent Actions and Interacting Effects

Javier A. Pinto
Depto. de Ciencia de la Computación,
Pontificia Universidad Católica de Chile,
Casilla 306, Santiago 22, CHILE.
Email: **jpinto@ing.puc.cl**

Abstract

Recently, several authors have proposed extensions to the language of the Situation Calculus in order to deal with concurrent actions. Two important problems have been recognized when modeling these actions. First, the *precondition interaction problem*, which arises when some action can or cannot be performed depending upon what other actions, if any, are performed concurrently. The second problem, the *effects interaction problem*, arises when a set of concurrently executed actions have effects that are different from the conjunction of the effects of each individual action. In this article, we discuss the second of these problems; namely, the *effects interaction problem.* First, we show that straightforward axiomatizations are not amenable to solutions based on policies of minimal change, due to the presence of unwanted minimal models. Then, we discuss alternative approaches to solve this problem. We advocate for an approach in which the interaction is decomposed along the temporal dimension, allowing the use minimal change and the elimination of spurious models.

1 INTRODUCTION

We explore the problem of modeling concurrency in theories of action within the framework of the Situation Calculus (McCarthy and Hayes 1969). Our work is based upon the premise that primitive actions are instantaneous. Actions with *durations* can be modeled as pairs of actions. Thus, the action "pour a cup of coffee" would be modeled with the start and the end of the pouring action. With each such action, one associates a fluent that is true while the action is in progress. As a consequence of treating actions with duration in this manner, problems like Allen's *door-latch* problem (Allen 1991) can be represented in terms of sequential actions. However, this way of modeling actions with durations, in a branching time language, introduces problems of its own. In particular, assume that one wants to model an action with duration that has an upper bound on its duration. To do this properly, one must ensure that in no path (in the branching structure) the action is started and not ended, or ended at a time that violates the duration upper bound. This problem is addressed in (Baier and Pinto 1998).

Primitive, instantaneous actions, when executed concurrently, may interact in two different ways. Firstly, there is *precondition interaction*, that arises when actions can or cannot be performed, depending upon whether other actions are performed concurrently. In general, this interaction arises when the actions require resources that might not be available if the actions are executed in parallel. For example, if there is one paint brush and two painters, then the actions of each painter painting interact, and they cannot be performed simultaneously.

Secondly, the effects of concurrently executed actions may also interact. In turn, this interaction can arise in two ways, as *cancellation* or as *synergy* (Pinto 1994). Cancellation is the phenomenon by which two actions cancel each other's effects when performed concurrently. For example, if a door is pushed and pulled at the same time and with the same force. Synergy arises when actions that are performed concurrently produce effects that neither action would have if performed in isolation. For example, lifting both ends of a table simultaneously has the effect of raising the table.

In this article, we focus our attention on the second type of interaction. A possible approach to deal with the *precondition interaction* problem is currently under development.

The article is organized as follows. In section 2 we present the formal foundations of the concurrent Situation Calculus. In section 3, we present the general structure of theories of action with concurrency. This structure is a straightforward extension of the style of axiomatization proposed by Reiter for theories without concurrency (Reiter 1991). In section 4 we define, in precise terms, the notion of *interacting effects* within the models of a theory of action. Later, in section 5, we show that the minimal change models of theories of action (as defined in section 3), in which there are interacting effects, do not have a single minimal model, leading to spurious minimal models. In section 6 we discuss approaches to solve this problem. In particular, we advocate for a solution in which the interacting effects are decomposed along the temporal dimension. This leads to axiomatizations in which *trigger* actions are used as intermediary agents of change. Finally, in section 8 we present our conclusions.

2 FORMAL FOUNDATIONS

We assume familiarity with the basics of the Situation Calculus (McCarthy and Hayes 1969), and with Reiter's monotonic solution to the frame problem (Reiter 1991). Our work is based on the concurrent Situation Calculus (Pinto 1994; Reiter 1996), which extends the Situation Calculus with a sort for concurrent actions. Concurrent actions are treated as sets of atomic actions.

The concurrent Situation Calculus is a second order language with sorts $\mathcal{A}$, $\mathcal{C}$, $\mathcal{S}$ and $\mathcal{D}$ for atomic actions, concurrent actions, situations and domain objects. The sort of the variables and constants used in the examples should be obvious from the context. There is a distinguished initial situation denoted with the constant S_0. The function *do* takes a concurrent action and a situation and yields another situation. Fluents are predicates that take one argument of type situation, and represent properties that are static within a situation, but that may change between situations. Also, we have the predicate *Poss*, which takes a concurrent action and a situation, and should hold when the concurrent action is executable in the situation. The foundational axioms for the concurrent Situation Calculus are:

$$(\forall\varphi).[\varphi(S_0) \wedge (\forall s,c)\ (\varphi(s) \supset \varphi(do(c,s)))] \supset (\forall s)\ \varphi(s), \tag{1}$$

$$do(c_1,s_1) = do(c_2,s_2) \supset c_1 = c_2 \wedge s_1 = s_2, \tag{2}$$

$$\neg s < S_0, \tag{3}$$

$$s < do(c,s') \equiv Poss(c,s') \wedge s \leq s'. \tag{4}$$

Concurrent actions are treated as sets of primitive actions. Thus, $\in$ is used as a relation between atomic actions and concurrent actions. We write $a \in c$ to mean that action a is part of the concurrent action c.

As discussed later, a domain axiomatization only includes axioms describing preconditions for atomic actions. For non-atomic actions, we have:

$$Poss(c,s) \equiv (\forall a)[a \in c \supset Poss(\{a\},s)]. \tag{5}$$

This axiom embodies the assumption that actions can be performed concurrently, if and only if all the component actions can be performed individually. In general, this is not correct, since there are actions that can be performed concurrently, but not individually and vice-versa.

In the following section we describe how to write theories of action based on the language of the concurrent Situation Calculus. Our approach is based upon Reiter's monotonic solution to the frame problem.

3 DOMAIN AXIOMATIZATION

A domain axiomatization is divided in several sets of axioms. First, we have a set T_d of axioms that only mention terms of sort $\mathcal{D}$. Also, we have a suitable set of unique names axioms (T_{una}, which we omit) for terms of sorts $\mathcal{A}$ and $\mathcal{D}$. The other sets of axioms are described below.

Action Precondition Axioms

The set T_{prec} of action precondition axioms for atomic actions, each of the form[1]:

$$Poss(\{a_l(\underline{\mathbf{x}})\},s) \supset \Phi_{a_l}(\underline{\mathbf{x}},s), \tag{6}$$

where $a_l(\underline{\mathbf{x}})$ is an $\mathcal{A}$ term, and $\Phi_{a_l}(\underline{\mathbf{x}},s)$ is a formula simple on s. A formula is simple on a situation term s_t, if it does not mention any situation terms, other than s_t, and does not quantify over s_t (Reiter 1991). Thus, preconditions for actions are formulated as necessary conditions for *Poss*. If we assume, as we will, that there are no *qualification state constraints* (Lin and Reiter 1994), we can apply Clark's completion to obtain a complete characterization of *Poss*.

[1]We write $\underline{\mathbf{x}}$ to denote a tuple of domain variables (i.e., variables of sort $\mathcal{D}$).

Example: Let us consider the following variant of the soup bowl example from (Gelfond, Lifschitz, and Rabinov 1991): There is a bowl filled with soup, the bowl can be lifted from either of two sides (left and right). If the bowl is lifted from one side only, then the soup spills. If the bowl is lifted from both sides simultaneously, then there is no spillage. To model this example, we have the action constants $Lift_l$, and $Lift_r$. There are three fluents $lifted_l$, $lifted_r$, and *spilling*.

The action precondition axioms for the actions in this domain are:

$$Poss(\{Lift_l\},s) \supset \neg lifted_l(s), \qquad (7)$$
$$Poss(\{Lift_r\},s) \supset \neg lifted_r(s), \qquad (8)$$

The assumption that these preconditions for performing actions are not only necessary, but also sufficient, leads to transforming each implication (7)-(8) into an equivalence (Clark's completion). A more general treatment of preconditions goes beyond the scope of this article[2].

Direct Effect Axioms

The set T_{eff} of effect axioms. These axioms are restricted to specify only the effects that are a direct consequence of the action, regardless of what other actions, if any, are performed concurrently.

Since *Poss* only takes concurrent actions as first parameter, these axioms have a slightly different syntactic form from Reiter's effect axioms (Reiter 1991). A positive direct effect axiom is of the syntactic form (9), while a negative direct effect axiom is of the form (10).

$$Poss(c,s) \wedge A \in c \wedge G_f^+(\underline{x},A,s) \supset f(\underline{x},do(c,s)), \qquad (9)$$
$$Poss(c,s) \wedge A \in c \wedge G_f^-(\underline{x},A,s) \supset \neg f(\underline{x},do(c,s)). \qquad (10)$$

Here, G_f^+ and G_f^- denote qualifications for the effects of an action on the fluent f.

Now, all the effect axioms can be compiled together in one positive and one negative general direct effect axiom of the forms (11) and (12).

$$Poss(c,s) \wedge \gamma_f^+(\underline{x},c,s) \supset f(\underline{x},do(c,s)), \qquad (11)$$
$$Poss(c,s) \wedge \gamma_f^-(\underline{x},c,s) \supset \neg f(\underline{x},do(c,s)). \qquad (12)$$

If there are no positive (or negative) effect axioms for a fluent f, then the formula $\gamma_f^+(\underline{x},c,s)$ (or $\gamma_f^-(\underline{x},c,s)$ for the negative case) is the atom *False*.

[2]For a more general treatment of qualifications for actions see (Lin and Reiter 1994).

Example (cont.): For the example, we have:

$$Poss(c,s) \wedge Lift_l \in c \supset lifted_l(do(c,s)),$$
$$Poss(c,s) \wedge Lift_r \in c \supset lifted_r(do(c,s)).$$

The Initial Situation

A set of axioms, T_{S_0}, characterizing the initial situation S_0.

Example (cont.): The initial situation for the example is characterized with:

$$\neg lifted_l(S_0) \wedge \neg lifted_r(S_0) \wedge \neg spilling(S_0).$$

Successor State Axioms

Reiter's monotonic solution to the frame problem leads to the derivation of a set of successor state axioms (one per fluent in the language). In the absence of state constraints (i.e., if T_{rc}, below, is empty), the successor state axioms for a fluent f would be derived from (11)-(12) along with the *explanation closure assumption* (Schubert 1990):

$$Poss(c,s) \supset [f(\underline{x},do(c,s)) \equiv \gamma_f^+(\underline{x},c,s) \vee f(\underline{x},s) \wedge \neg\gamma_f^-(\underline{x},c,s)]. \qquad (13)$$

These axioms state necessary and sufficient conditions for fluents to change after some possible action is performed. This solution was originally proposed for theories of actions without concurrency and without state constraints. This work was extended to theories in which state constraints are present (Lin and Reiter 1994; Pinto 1994). These results show that Reiter's solution is not always applicable. In particular, in (Pinto 1994), it is shown that the approach is not directly applicable when there are constraints such as (15), below. If Σ denotes a theory formed with the previous sets of axioms, we will write $\mathcal{R}(\Sigma)$ to denote the theory Σ extended with the successor state axioms.

Ramification State Constraints

A set T_{rc} of ramification state constraints (Lin and Reiter 1994). In general, ramification constraints are of the form:

$$\Psi(\underline{x},s), \qquad (14)$$

where $\Psi(\underline{x},s)$ is a simple formula. We assume that constraints are written as clauses. Also, it might be convenient to write these clauses as implications.

Example (cont.): For this domain we have:

$$spilling(s) \equiv [\neg lifted_l(s) \wedge lifted_r(s) \vee lifted_l(s) \wedge \neg lifted_r(s)] \qquad (15)$$

Thus, this constraint states that *spilling* holds if and only if a single side of the bowl is lifted.

4 INTERACTING EFFECTS

In the following discussion, we assume a theory of action Σ_{rc}, composed of axioms T_d, T_{una}, T_{prec}, T_{eff}, T_{rc} and T_{S_0}, as in the previous section, in addition to the foundational axioms of section 2.

At an informal level, we say that there is interaction between the effects of actions when there is some concurrent action that either:

- has effects that are not present when the sub-actions are performed individually (synergy), or
- does not have effects that some of its sub-actions have when performed individually (cancellation).

The example with the bowl of soup is a case where there is cancellation of effects. Performing $Lift_l$ together with $Lift_r$ cancels the *spilling* effect that each action would have, if performed by itself.

In order to simplify the following discussion, we assume that fluents are propositional (i.e., they only take a situation argument). However, it is straightforward to extend the analysis below to the more general case.

In what follows, we use σ_a, σ_c and σ_s to denote assignment functions from free variables of sort $\mathcal{A}$, $\mathcal{C}$ and $\mathcal{S}$ to objects of the corresponding sorts. Before we can formalize the notion of effect interaction, we need the following:

Definition 4.1 (Model Similarity) *Let Σ_{rc} be a theory of action, and σ_s be an assignment function from free situation variables to situations. We say that two models, $\mathcal{M}$ and $\mathcal{M}'$ of Σ_{rc} are similar with respect to $\sigma_s(s)$ iff:*

- *$\mathcal{M}$, $\mathcal{M}'$ have the same universe, and differ only in their interpretation of fluents,*
- *for any fluent f:*

$$\mathcal{M},\sigma_s \models f(s)$$

if and only if

$$\mathcal{M}',\sigma_s \models f(s).$$

Definition 4.2 (Synergy) *Synergistic effects arise in a set $\mathbf{M}_{\Sigma_{rc}}$ of models of a theory of action Σ_{rc}, with respect to a fluent f, when there is a model $\mathcal{M}$ in $\mathbf{M}_{\Sigma_{rc}}$, and assignment functions σ_s and σ_c such that:*

1. *In the model $\mathcal{M}$, there is a concurrent action that makes the fluent f change in some situation:*

$$\mathcal{M},\sigma_s,\sigma_c \models Poss(c,s) \wedge ab(f,s,c).$$

where $ab(f,s,c)$ is a shorthand for:

$$f(s) \equiv \neg f(do(c,s)).$$

2. *For any model $\mathcal{M}'$ in $\mathbf{M}_{\Sigma_{rc}}$, similar to $\mathcal{M}$ with respect to $\sigma_s(s)$,*

$$\mathcal{M}',\sigma_s,\sigma_c \models ab(f,s,c).$$

3. *For some model $\mathcal{M}''$ in $\mathbf{M}_{\Sigma_{rc}}$ similar to $\mathcal{M}$ with respect to $\sigma_s(s)$, there is no assignment σ_a, such that:*

$$\mathcal{M}'',\sigma_s,\sigma_c,\sigma_a \models a \in c \wedge ab(f,s,\{a\}).$$

Intuitively, condition 1 states that there is a model and situation $\sigma_s(s)$ in which f changes when the action $\sigma_c(c)$ is performed. Furthermore, condition 2 states that f must change in the models in $\mathbf{M}_{\Sigma_{rc}}$. I.e., there is no model in $\mathbf{M}_{\Sigma_{rc}}$, similar to the first one with respect to $\sigma_s(s)$, that does not change f when the concurrent action is performed. Condition 3 states that synergy arises. I.e., the change arises with the concurrent action $\sigma_c(c)$, but that there is no single action in $\sigma_c(c)$ that forces f to change.

Definition 4.3 (Cancellation of Effects) *Cancellation of effects arise in a set $\mathbf{M}_{\Sigma_{rc}}$ of models of a theory of action Σ_{rc}, with respect to a fluent f, when there is a model $\mathcal{M}$ in $\mathbf{M}_{\Sigma_{rc}}$, and assignment functions σ_s, σ_a and σ_c such that:*

1. *In the model $\mathcal{M}$, the fluent f does not change when some concurrent action is performed:*

$$\mathcal{M},\sigma_s,\sigma_c \models Poss(c,s) \wedge \neg ab(f,s,c).$$

2. *For any model $\mathcal{M}'$ in $\mathbf{M}_{\Sigma_{rc}}$, similar to $\mathcal{M}$ with respect to $\sigma_s(s)$,*

$$\mathcal{M}',\sigma_s,\sigma_c \models \neg ab(f,s,c).$$

3. *For some model $\mathcal{M}''$ in $\mathbf{M}_{\Sigma_{rc}}$ similar to $\mathcal{M}$ with respect to $\sigma_s(s)$:*

$$\mathcal{M}'',\sigma_s,\sigma_c,\sigma_a \models a \in c \wedge ab(f,s,\{a\}).$$

Intuitively, condition 1 states that there is a model in $\mathbf{M}_{\Sigma_{rc}}$ and situation in which f does not change when some specific concurrent action is performed. On the other hand 2, ensures that f must not change. I.e., there is no model in $\mathbf{M}_{\Sigma_{rc}}$, similar to the first one with respect to $\sigma_s(s)$, that changes f when the concurrent action is performed. Condition 3 states that an effect has been canceled. I.e., there is a primitive action, that is included in the concurrent action, mentioned above, after which the change arises.

Example (cont.): As mentioned before, the bowl of soup example is a case of cancellation. Indeed, if Σ_{bowl} is the theory of action for that example:

$$\Sigma_{bowl} \models Poss(\{Lift_l, Lift_r\}, S_0) \wedge Lift_l \in \{Lift_l, Lift_r\},$$
$$\Sigma_{bowl} \not\models ab(spilling, S_0, \{Lift_l\}),$$
$$\Sigma_{bowl} \not\models \neg ab(spilling, S_0, \{Lift_l\}),$$

and

$$\Sigma_{bowl} \models \neg ab(spilling, S_0, \{Lift_l, Lift_r\}).$$

From here, it is straightforward to verify that the conditions for cancellation are present in the set of first order models of Σ_{bowl} (they are also present in the minimal change models of Σ_{bowl}.

Observations

The definitions for synergy and cancellation given above are based on comparing the results of performing a concurrent action c, and the results of performing some primitive sub-action a, where $a \in c$. However, more general definitions can be given by comparing the results of performing a concurrent action c and the results of performing some other concurrent action c', such that $c' \subsetneq c$. It is not hard to rewrite the definitions. But, we kept the definitions presented here for simplicity of exposition.

Notice that, as motivated by many examples, synergy and cancellation are properties that we would like our theories to exhibit. Therefore, after solving the frame problem, we would like the resulting theories to exhibit these properties as well. The result that we show, is that if the minimal models of a theory of action exhibit synergy or cancellation, then the theory must admit more than one minimal model. Thus, there might be spurious minimal models.

5 MINIMAL CHANGE POLICIES

In the previous section we presented a formal characterization of the interaction between the effects of actions. Up until now, we have assumed that the theory of action includes axioms describing necessary effects, along with axioms establishing ramification state constraints. It is well known that these axioms are not enough to characterize change, and that some approach to derive non-change should also be considered, in the form of *frame axioms* or by some other means. A common approach to complete the specification of the theory of action is to incorporate the assumption of minimal change. This minimal change policy can be formalized with the use of circumscription (McCarthy 1986), as in (Baker 1989; Shanahan 1997). Also, in some cases this assumption can be realized by using *explanation closure* (Schubert 1990), which leads to the formulation of successor state axioms (Reiter 1993). This latter approach results in a theory in which the effects of actions are completely characterized, given complete knowledge about the state in which they are performed.

Here, we argue that an approach based on minimal change cannot properly characterize change in theories of the form discussed in the previous section, in which there is interaction between the effects of actions.

The following definition (based on (Lin and Reiter 1994)) characterizes minimal change models:

Definition 5.1 (Minimal Model) *A model $\mathcal{M}$ of a theory of action Σ_{rc} is minimal relative to $\sigma_s(s)$ iff there is no other model $\mathcal{M}'$ of Σ_{rc}, similar to $\mathcal{M}$ with respect to $\sigma_s(s)$, such that:*

- *For any assignment σ_c, and any fluent f:*

$$\begin{array}{rl} \textit{if} & \mathcal{M}, \sigma_s, \sigma_c \models Poss(c,s) \wedge \neg ab(f,s,c), \\ \textit{then} & \mathcal{M}', \sigma_s, \sigma_c \models Poss(c,s) \wedge \neg ab(f,s,c). \end{array}$$

- *There is an assignment σ_c, and a fluent f, such that:*

$$\begin{array}{rl} & \mathcal{M}, \sigma_s, \sigma_c \models Poss(c,s) \wedge ab(f,s,c), \\ \textit{but} & \mathcal{M}', \sigma_s, \sigma_c \models Poss(c,s) \wedge \neg ab(f,s,c). \end{array}$$

Also, we say that a model is minimal, if it is a minimal model relative to situation $\sigma_s(s)$, for any assignment σ_s.

The following definition is inspired in the notion of well-foundedness defined in (Lifschitz 1994):

Definition 5.2 (Well Foundedness) *Let Σ_{rc} be a theory of action, we say that Σ_{rc} is well founded if for any model $\mathcal{M}'$ and assignment function σ_s there exists a model $\mathcal{M}$ similar to $\mathcal{M}'$ with respect to a situation $\sigma_s(s)$, which is also minimal relative to $\sigma_s(s)$.*

Theorem 5.1 *Let Σ_{rc} be a well founded theory of action, composed of the foundational axioms of section 2, along with axioms T_d, T_{una}, T_{prec}, T_{eff}, T_{rc} and T_{S_0}, as in section 3. If cancellation of effects or synergy arise in the set of minimal models of Σ_{rc}, then Σ_{rc} does not have a unique minimal model.*

Proof Let us assume that Σ_{rc} is a theory of action in which cancellation of effects arise (the proof for synergy is similar). Thus, let $\mathcal{M}''$, σ_s, σ_c and σ_a be as in definition 4.3, and let us assume that $\mathcal{M}''$ is minimal. In this model, we have:

$$\mathcal{M}'', \sigma_s, \sigma_c, \sigma_a \models Poss(c,s) \wedge a \in c \wedge ab(f,s,\{a\}),$$

and

$$\mathcal{M}'', \sigma_s, \sigma_c \models \neg ab(f,s,c).$$

Now, we can build a model $\mathcal{M}'$ similar to $\mathcal{M}''$ such that for any fluent g:

$$\begin{aligned} & \mathcal{M}', \sigma_s, \sigma_c, \sigma_a \models g(do(\{a\},s)), \\ \text{iff} \quad & \mathcal{M}'', \sigma_s, \sigma_c, \sigma_a \models g(do(c,s)). \end{aligned}$$

This assignment satisfies the state constraints, otherwise $\mathcal{M}''$ would not satisfy the state constraints either. Given this construction:

$$\mathcal{M}', \sigma_s, \sigma_c, \sigma_a \models \neg ab(f,s,\{a\}).$$

From the well-foundedness of Σ_{rc}, it follows that there will be a minimal model $\mathcal{M}^{m'}$ relative to $\sigma_s(s)$, in which the abnormality is not present.

An alternative to this proof considers the necessary existence of ramification constraints, responsible for the interaction between effects. For illustrative purposes, we provide a sketch for this alternative proof. We can differentiate the direct effects of actions from the indirect effects, derivable as ramifications of the direct effects and the state constraints in T_{rc}. Without the set T_{rc} of ramification constraints, there can be no interacting effects. Indeed, it is easy to see that, in this case, the change that is produced by a concurrent action is simply the conjunction of the changes produced by the individual actions. Therefore, if f is the fluent that changes due to synergistic effect interaction (the argument for cancellation is similar), then T_{rc} along with T_d and T_{una}, must entail a formula of the form:

$$l_1(s) \wedge \ldots \wedge l_n(s) \supset f(s),$$

where $l_1(s) \ldots l_n(s)$ denote fluent literals. Also, we have assumed that f changes positively. Therefore, after the concurrent action is performed, all l_i fluent literals are made true, and each l_i is a direct effect of a primitive action in the synergistic concurrent action. From here it follows that, given the definition of synergy, some action has some, but not all, the l_i's as direct effects. Therefore, this action would have a disjunction of literals as a ramification, without having any of its disjuncts as a direct effect. From here it follows that there will be several minimal models.

Example (cont.): According to the previous result, the soup bowl example cannot have a single minimal model. Indeed, one minimal model is the intended one. That is, the model in which $lifted_l$ is true in a situation $do(c,S_0)$ if and only if $Lift_l$ belongs to c. For instance, we would have a model in which:

$$\begin{aligned} & spilling(do(\{Lift_l\},S_0) \wedge lifted_l(do(\{Lift_l\},S_0)) \wedge \\ & \neg lifted_r(do(\{Lift_l\},S_0)). \end{aligned}$$

However, there is also a minimal model in which:

$$\begin{aligned} & \neg spilling(do(\{Lift_l\},S_0) \wedge lifted_l(do(\{Lift_l\},S_0)) \wedge \\ & lifted_r(do(\{Lift_l\},S_0)). \end{aligned}$$

Thus, in this second model, lifting the left side of the bowl has the effect of lifting both sides. It is not difficult to verify that there is a minimal model in which this holds. On the other hand,

$$\begin{aligned} & \neg spilling(do(\{Lift_l, Lift_r\},S_0) \wedge \\ & lifted_l(do(\{Lift_l, Lift_r\},S_0)) \wedge lifted_r(do(\{Lift_l, Lift_r\},S_0)). \end{aligned}$$

is true in all minimal models.

6 DEALING WITH INTERACTING EFFECTS

Why is the presence of interacting effects a problem? If one wants to build a theory of actions in the presence of concurrency, it would be inconvenient to specify the direct effects of all possible concurrent actions; in particular, modularity would be lost if every new action that is incorporated in the model required a reformulation of the effect axioms for concurrent actions. Therefore, it would be desirable that these effects be derivable from the direct effects of the atomic actions that are performed as part of the concurrent action. However, as the previous result states, if the effects of actions interact, an axiomatization of the style proposed in section 3, along with a minimal model semantics, fails to characterize the intended models.

How can we proceed? In situations like the one illustrated by the soup bowl example, it is very clear which model is the intended one. At this point, there are a few alternatives. Clearly, one can argue that there is not enough information built into the theory in order to single out the *right* model. In fact, one can add *causal* information, and state that lifting one side of the bowl cannot make the other side move, but can cause a spill. This causal information can be used to filter unwanted models. Therefore, in this approach the theories of action would be extended with explicit *causal axioms*. This extension would need to be accompanied by a more sophisticated minimal model semantics. Approaches based on this idea have been used in order to solve the ramification problem. For instance, see (Lin 1995; Thielscher 1997).

A second alternative is to use a *categorization based* approach. For example, we can define a stratification for the fluents. In the example, we could define $lifted_l$ and $lifted_r$ to be in one stratum, and *spilling* can be placed in another stratum. Then, we can prefer models where persistence in one stratum is preferred to persistence in another. In this case, each stratum would correspond to a separate category. This categorization of fluents is also used to deal with the ramification problem. For example, in (Lifschitz 1990; Brewka and Hertzberg 1993; del Val and Shoham 1993).

A third approach, which we pursue here, is based on the observation that the proposed theory fails to characterize the nature of the reality being modeled. Thus, we propose an alternative approach to modeling situations in which the effects of the individual actions interact. This approach is based on the observation that indirect effects of actions may arise because of the existence of intervening events. For instance, if one side of the bowl of soup is lifted, then there is an event, *Spills*, which is triggered by the lifting action. The *Spills* event is triggered in an intermediate state in which only one side of the bowl is lifted, and yet *spilling* is not true. Therefore, modeling the problem in this form implies changes in the way the problem is axiomatized. In particular, because some of the state constraints originally proposed might be violated. This approach has the following advantages:

- In many cases, the nature of the problem that is modeled is such that the intermediate states exist, probably with a very short time span. For instance, in the soup bowl example, it is not unreasonable to consider the existence of an intervening state where *spilling* is not true and only one side of the bowl is lifted. This intervening state will have a very short duration and will trigger the event *Spills*. Therefore, introducing an intermediate state seems to be a more accurate portrayal of the phenomenon.
- Minimal change approaches are still applicable. The problem that was described in the previous section does not arise. In fact, constraints responsible for interaction are replaced by a triggering axiom, (see below) and new effect axioms. In this approach, minimization of change is only meant to deal with the frame problem.
- The conditions on triggered events are expressed in terms of the truth values of fluents, as opposed to the actions that are performed. For example, independently of what actions are involved, if only one side of the bowl is lifted, and *spilling* is not already true, then *Spills* will occur. This is independent of what other actions might be present in the model. Therefore, the axiomatization remains modular.

Natural Actions and Triggers

In order to handle triggers, we extend the language of the Situation Calculus with the notion of *natural actions* (Pinto 1994; Reiter 1996). Natural actions are not regular *agent actions*. Instead, they are actions that arise due to the nature of the world. One characteristic of these actions is that they must occur (i.e., they are triggered) whenever they are possible. Thus, we need:

$$(\forall a,s).\ Poss(\{a\},s) \land natural(a) \supset occurs(a,s). \quad (16)$$

I.e., if a natural action is possible, it must occur. There are several ways in which occurrences can be modeled in the Situation Calculus. One approach is to consider that there is a single branch or narrative line in the tree of situations, such that the actions that occur fall in that branch (Miller and Shanahan 1994; Pinto and Reiter 1993). An alternative is to consider occurrences as constraints that limit the situations that can be considered legal, as in (Pinto 1998). Here, we use a simplified version of the latter approach. We define the predicate *legal* for situations, such that:

$$\begin{aligned}(\forall s).\ legal(s)\ &\equiv (s \geq S_0 \land \\ &(\forall s',c,a)\ [s' < s \land occurs(a,s') \land \\ &do(c,s') \leq s \supset a \in c]).\end{aligned} \quad (17)$$

Thus, a situation is considered legal if all the actions that lead from S_0 to it are possible (expressed with the $s \geq S_0$ literal), and if all occurrences within the branch $S_0 - s$ are present in the branch. Therefore, to reason with occurrences, we focus on the situations that are *legal*.

Notice that the predicate *occurs*, as well as the predicate *Poss*, is not a fluent, since it is not used to describe the *state* of a situation. Rather, it is used to constraint the dynamics of the world. Naturally, the state of a situation might determine the truth of *occurs*, affecting, indirectly, the dynamics of the world.

Later on, we will make use of the notion of situation stability. Situations in which natural actions are possible are not considered to be stable, thus, motivating the introduction of the predicate $stable \subseteq \mathcal{S}$ with the definition:

$$stable(s) \equiv \neg(\exists a).natural(a) \wedge Poss(\{a\},s). \tag{18}$$

Furthermore, we introduce the predicate $leads \subseteq \mathcal{C} \times \mathcal{S} \times \mathcal{S}$ with the definition:

$$\begin{aligned} leads(c,s,s') \equiv\ & s < do(c,s) \leq s' \wedge stable(s') \wedge \\ & ((\forall s_i)\ s < s_i < s' \supset \neg stable(s_i)) \wedge \\ & ((\forall c_i,s_i,a_i)\ do(c,s) < do(c_i,s_i) \leq s' \wedge a \in c_i \supset \\ & \qquad natural(a)). \end{aligned} \tag{19}$$

This latter predicate relates a situation s and an action a to the *stable* situation that results after all natural actions triggered by a in s are performed. The third conjunct on the right side of the equivalence requires that all intermediate situations be unstable. The last conjunct establishes that the resulting stable situation be reached by performing only natural actions.

Triggers and interacting effects.

As mentioned in the sketch of the proof of theorem 5.1, the interactions between effects arise because of the presence of ramification constraints. Our proposal is based on the hypothesis that the ramification constraints can be decomposed along the temporal dimension. What this means is that some constraints hide a process that can be modeled at a higher level of detail. For instance, a constraint of the form:

$$l_1(s) \wedge l_2(s) \supset l(s). \tag{20}$$

where l_1, l_2 and l represent some fluent literals, might be used to model a causal relationship between these literals. Since causality cannot be captured with implication alone, problems arise. One possibility for decomposing this type of constraint is to introduce new events as intermediate *causes*. Thus, instead of writing a constraint like (20), we propose to introduce some intermediate natural action A_l in the language, and add the following axioms:

$$natural(A_l), \tag{21}$$
$$l_1(s) \wedge l_2(s) \wedge \neg l(s) \supset Poss(\{A_l\},s), \tag{22}$$
$$Poss(c,s) \wedge A_l \in c \supset l(do(\{A_l\},s)). \tag{23}$$

Furthermore, it is convenient to assume that the necessary conditions for the truth of *Poss* are also sufficient. Thus, converting sets of implications like (22) into equivalences.

Under these conditions, the interaction of effects, due to the original state constraints, is converted into a triggered effect. Synergy or cancellation due to state constraints can be handled in this manner in many important cases, as described in the next section.

Example (cont.): In the soup bowl example, we can replace the ramification constraints with the precondition axioms:

$$\begin{aligned} &\neg lifted_l(s) \wedge lifted_r(s) \wedge \neg spilling(s) \supset Poss(\{Spills\},s), \\ &lifted_l(s) \wedge \neg lifted_r(s) \wedge \neg spilling(s) \supset Poss(\{Spills\},s), \\ &lifted_l(s) \wedge lifted_r(s) \wedge spilling(s) \supset Poss(\{Nspills\},s), \\ &\neg lifted_l(s) \wedge \neg lifted_r(s) \wedge spilling(s) \supset Poss(\{Nspills\},s), \end{aligned}$$

Along with the requirement:

$$natural(Spills) \wedge natural(Nspills),$$

and the effect axioms:

$$\begin{aligned} &Poss(c,s) \wedge Spills \in c \supset spilling(do(c,s)), \\ &Poss(c,s) \wedge Nspills \in c \supset \neg spilling(do(c,s)). \end{aligned}$$

Here, we have added two different natural actions, one that results in spillage, and the other that stops it.

Our proposal to decompose ramification state constraints in the temporal dimension, by the introduction of new natural actions with their derived triggered effects, is based on the following observations:

- In our everyday lives we function under the *common-sense* assumption that actions are deterministic. Non-determinism, in this context, arises because of lack of knowledge of the state of the world, or for lack of control regarding the actions that are performed.
- In physical domains, such as the bowl with soup example, one can accurately predict the direct effects of actions.
- Indirect effects of actions arise because the action creates the conditions for these indirect effects to arise. In general, these effects can be considered to arise subsequent to the performance of the action.

Our view is based on the last observation. Synergy or cancellation arises because of the indirect effects of one or more actions. Thus, this approach is applicable in settings where these observations hold.

The applicability of this approach can also be judged by comparing it to other approaches to formalize the same phenomena. In the following section, we propose the alternative of treating the set T_{rc} of ramification constraints as *causal rules*, inspired by the work of Lin (1995, 1998). Then, we show how these causal rules can be translated into natural actions and triggered effects in a mechanical manner.

7 CAUSAL RULES

7.1 PROPOSITIONAL FLUENTS

In this section we discuss some preliminary results regarding the applicability of this approach to handle effects interactions in theories with concurrent actions. For simplicity, we assume that all fluents are propositional. Thus, fluents are predicates that only take a single argument of sort $\mathcal{S}$.

We start with a theory of action Σ_{rc}, as before. The theory includes the set T_b of axioms (1)-(5), along with a set T_d of domain specific axioms (that only mention terms of type $\mathcal{D}$), a set T_{prec} of action precondition axioms of the form (6), a set T_{eff} of direct effect axioms like (11)-(12), a set T_{rc} of ramification state constraints of the form (14), and a set of axioms T_{S_0} that describe the initial situation (a simple formula applied to S_0).

As mentioned earlier, a set of state constraints are not enough to characterize change in the presence of concurrent actions and interaction between effects. What we need is more refined *causal* information. We will assume that each state constraint in the set T_{rc} is written as a *causal* implication.

Now, we take a theory of action Σ_{rc} and mechanically derive a new theory Σ_{nat}. The steps that are followed to derive this new theory are the following:

1. Each axiom in T_{rc} should be written as a *causal rule*, such as:

$$l_1(s) \wedge \ldots \wedge l_n(s) \supset l(s). \tag{24}$$

   The implication symbol will be interpreted as a *causality operator*, and not as logical implication. Therefore, the syntactic form of these axioms is very important. We should use another operator for causality, and rewrite the rules with this new operator. Given the space limitations, we are not able to pursue this in detail. Instead, we appeal to the reader's understanding.

   How this step is achieved depends crucially on the domain of application. All causal rules of the form (24), with the same literal on the right-hand side can be compiled together in a single causal rule of the form:

$$\Phi_l(s) \supset l(s). \tag{25}$$

   For notational convenience, we write $T_{rc}(s)$ as the conjunction of all the formulas in T_{rc}, leaving s free. We will assume that there are no axioms in T_{eff} in which the literal l is falsified.

   Our interpretation of these causal rules corresponds roughly to what Lin has called *dynamic causal rules* (Lin 1998).

   These causal rules will not be part of the resulting theory Σ_{nat}, but are used in the steps below.

2. The original language is extended with one new action constant per causal rule. The T_{prec} axioms are extended with action precondition axioms of the form (one axiom per causal rule (25)):

$$\Phi_l(s) \supset Poss(\{A_l\}, s),$$

   where A_l is a new action constant. Let T^n_{prec} denote the extended set. Also, T_{una} is extended for these new action constants, leading to T^n_{una}

3. The axioms T_{eff} are extended with a new effect axiom for each new action A_l:

$$Poss(c,s) \wedge A_l \in c \supset l(do(\{A_l\}, s)).$$

   Let T^n_{eff} denote the extended set of axioms.

4. Σ_{rc} is extended with a new set of natural axioms, for each new action symbol added to the language in the previous step, we add:

$$natural(A_l).$$

   We will assume that these are the only natural actions. This assumption can be formalized by a minimization of the extension of this predicate. Given the simple form of these axioms, we can simply write: $natural(a) \equiv a = A_{l1} \vee a = A_{l2} \ldots$. Let T_{nat} be a singleton set with this complete specification for *natural*.

5. The axioms T_{S_0} are extended with a $T_{rc}(S_0)$, that is, for each state constraint $(\forall s)\ \Psi(s)$ in T_{rc}, we add $\Psi(S_0)$ as an axiom. Let $T^n_{S_0}$ denote the extended set of axioms about S_0.

6. The theory Σ_{nat} is given by the foundational axioms of the situation calculus, axioms about natural actions (16)-(19), and T_d, T^n_{una}, T^n_{eff}, T^n_{prec}, T_{nat} and $T^n_{S_0}$.

The theory Σ_{nat} is of a form such that we can apply Reiter's approach to derive a monotonic solution to the frame problem. Indeed, except for the axioms about natural actions, the structure of the axiomatization is a standard one. Furthermore, the axioms about natural actions do not interfere with the changes that are direct consequence of actions. Therefore, we can obtain successor state axioms for all fluents of the domain. Let $\mathcal{R}(\Sigma_{nat})$ be the theory obtained from Σ_{nat} following Reiter's approach.

7.2 SOME PROPERTIES

Previously, we described how Σ_{rc} can be transformed into a theory $\mathcal{R}(\Sigma_{nat})$. What properties does the resulting theory have with respect to the original one? First, does the resulting theory entail the state constraints for legal situations? The answer is no, thus:

$$\mathcal{R}(\Sigma_{nat}) \not\models (\forall s).\ legal(s) \supset T_{rc}(s).$$

$T_{rc}(s)$ is the conjunction of the constraints (26)-(27) applied to s. For example, consider the theory Σ_{rc}:

T_{prec}: $\quad Poss(\{A\},s),$

T_{eff}: $\quad Poss(\{A\},s) \supset f_1(do(\{A\},s)),$

T_{rc}: $\quad f_1(s) \supset f_2(s), \qquad (26)$

$\quad f_2(s) \supset f_3(s), \qquad (27)$

T_{S_0}: $\quad \neg f_1(S_0) \wedge \neg f_2(S_0) \wedge \neg f_3(S_0).$

The theory that results from applying the previous scheme leads to a new theory in which the new action constants A_{f2} and A_{f3} are introduced. After deriving the successor state axioms, we obtain a theory $\mathcal{R}(\Sigma_{nat})$ that includes the following axioms:

$$\begin{aligned}
&natural(a) \equiv a = A_{f2} \vee a = A_{f3},\\
&Poss(\{A\},s),\\
&Poss(\{A_{f2}\},s) \equiv f_1(s) \wedge \neg f_2(s),\\
&Poss(\{A_{f3}\},s) \equiv f_2(s) \wedge \neg f_3(s),\\
&Poss(c,s) \supset f_1(do(c,s)) \equiv A \in c \vee f_1(s),\\
&Poss(c,s) \supset f_2(do(c,s)) \equiv A_{f2} \in c \vee f_2(s),\\
&Poss(c,s) \supset f_3(do(c,s)) \equiv A_{f3} \in c \vee f_3(s),\\
&\neg f_1(S_0) \wedge \neg f_2(S_0) \wedge \neg f_3(S_0).
\end{aligned}$$

Clearly:

$$\mathcal{R}(\Sigma_{nat}) \models f_1(do(\{A\},S_0)) \wedge \neg f_2(do(\{A\},S_0)).$$

Thus, violating the original state constraint (26). However, we do have the following:

$$\mathcal{R}(\Sigma_{nat}) \models (\forall s).\ leads(\{A\},S_0,s) \supset T_{rc}(s).$$

In general:

Proposition 7.1 *Let Σ_{rc} be a theory of action, as in the previous sections, and let $\mathcal{R}(\Sigma_{nat})$ be the theory that results from the application of the steps described earlier. Then:*

$$\mathcal{R}(\Sigma_{nat}) \models (\forall s).\ stable(s) \wedge legal(s) \supset T_{rc}(s).$$

The proof of this proposition is fairly direct. If the state constraints were violated in a given situation, then the situation would not be a stable one, prompting for some natural action to occur in it.

If Σ_{rc} is a theory of action, and if one wants to know whether some fluent f holds after performing an action A in some situation S, one simply tries to find out whether:

$$\Sigma_{rc} \models S_0 < S \wedge f(do(\{A\},S)).$$

However, in theories with natural actions, such as $\mathcal{R}(\Sigma_{nat})$, this question becomes:

$$\mathcal{R}(\Sigma_{nat}) \models (\forall s).\ leads(\{A\},S,s) \supset f(do(\{A\},s)).$$

Thus, we are interested in *stable* changes. Now, we can define abnormality with respect to stable situations as follows:

$$\begin{aligned}
&ab^+(f,s,c) =_{\mathbf{def}}\\
&\quad stable(s) \wedge leads(c,s,s') \wedge [f(s) \equiv \neg f(s')]
\end{aligned}$$

Another property of $\mathcal{R}(\Sigma_{nat})$, in relation to Σ_{rc}, is that there is a natural correspondence between the models of $\mathcal{R}(\Sigma_{nat})$ and the minimal models of Σ_{rc}. If $\mathcal{M}^+$ is a model of $\mathcal{R}(\Sigma_{nat})$, then there must be a minimal model $\mathcal{M}$ of Σ_{rc}, such that for each abnormality ab^+ in $\mathcal{M}^+$ there is an abnormality ab in $\mathcal{M}$.

In order to know whether some fluent becomes true after performing some action in an stable situation, we must find an stable situation after the action is performed. If fluents are propositional and finite in number, and the theory is consistent, it is not hard to show that this stable situation must exist. However, this is not true for theories with non-propositional fluents.

7.3 NON-PROPOSITIONAL FLUENTS

If the language has non-propositional fluents, the existence of a stable situation, after an action is performed, is not guaranteed. Indeed, assume a theory Σ_{rc} in which T_{rc} has a state constraint such as:

$$f(x,s) \supset f(s(x),s),$$

along with an A and an effect axiom:

$$Poss(c,s) \wedge A \in c \supset f(x,do(c,s)).$$

and an initial situation specification with:

$$(\forall x)\ \neg f(x,s).$$

It is not difficult to build a theory Σ_{rc} with these axioms such that $\mathcal{R}(\Sigma_{nat})$ will have models in which $leads(A,S_0,s)$ is identically false.

However, there are many cases in which we can guarantee the existence of stable states after performing any action. For instance, we can take stratified sets of causal rules. The definition of stratification is similar to the one used in logic programming (Apt, Blair, and Walker 1987). Thus, we can define a numbering scheme in which each fluent receives a number. A set T_{rc} is considered stratified if there is a numbering scheme in which, for each causal rule, the numbers assigned to the fluents in the antecedent, are strictly smaller that the numbers assigned to the fluent in the consequent. It is not difficult to show that if the set T_{rc} is stratified, a translation would yield a theory in which stability is guaranteed.

8 CONCLUSION

In this article we have discussed the problem of formalizing theories of action and change for worlds in which instantaneous actions can be executed concurrently. We showed that axiomatizations in which the effects of concurrently executed actions interact, due to the presence of state constraints, admit multiple minimal models. This result holds for theories in which there is a single intended minimal model. The existence of this unwanted models leads to the conclusion that there is relevant information about the dynamics of the world which is not present in the original axiomatization.

In order to deal with this problem, we propose the use of natural actions as primitive elements in the axiomatization of physical domains. Thus, an axiomatizer should directly encode domain knowledge using axioms about natural actions, as opposed to writing state constraints, from which the axioms about natural actions could be mechanically obtained. However, the mechanism proposed allows one to have a better understanding of the applicability of the approach. We believe that an approach to modeling based on the characterization of triggered or natural actions should be more natural (sic). The advantages of this approach are:

- We can use a standard minimal-model semantics. In particular, this allows to use Reiter's monotonic approach to deal with the frame problem.
- The axiomatization is modular with respect to the modeling of actions. One does not need to specify what are the effects of actions in combination with other actions that might be performed concurrently. Instead, intervening actions are triggered depending on the state of the world (independently of the manner in which the state was reached).

Also, we discuss an approach to deal with interacting effects based on the interpretation of ramification state constraints as *causal rules*. This approach depends crucially upon the syntactic form in which the state constraints are written. Thus, the results that are obtained are different depending upon how the rules are written. For instance, the constraint $f_1(s) \supset f_2(s)$ would lead to a theory of action that would not be equivalent to one in which the constraint were written as $f_2(s) \supset f_1(s)$. However, assuming that the rules are written with their intended causal interactions in mind, they can be translated to the approach based on triggered events.

The research that is presented in this article is closely related to research on ramification and causality. In particular, there are interesting relationships to the work of Lin (1995, 1998), Thielscher (1997), Baral and Gelfond (1997) and Gustafsson and Doherty (1996). For future research, we would like to explore this relationship.

Acknowledgements

This work has been partly funded by the Fondecyt project number 197-1304, and by the PRAXIS XXI Project 2/2.1/MAT/262/94 SitCalc.

References

Allen, J. (1991). Planning as Temporal Reasoning. In J. Allen, R. Fikes, and E. Sandewall (Eds.), *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, pp. 3–14. Morgan Kaufmann Publishers, Inc.

Apt, K., H. Blair, and A. Walker (1987). Towards a Theory of Declarative Knowledge. In J. Minker (Ed.), *Foundations of Deductive Databases and Logic Programming*, pp. 89–148. Morgan Kaufmann.

Baier, J. and J. Pinto (1998). Non-instantaneous Actions and Concurrency in the Situation Calculus. In preparation.

Baker, A. (1989, May). A Simple Solution to the Yale Shooting Problem. In H. L. R. Brachman and R. Reiter (Eds.), *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, pp. 11–20. Morgan Kaufmann Publishers, Inc.

Baral, C. and M. Gelfond (1997). Reasoning About Effects of Concurrent Actions. *Journal of Logic Programming 31*, 85–118.

Brewka, G. and J. Hertzberg (1993). How to do things with worlds: on formalizing actions and plans. *Journal of Logic and Computation 3*, 517–532.

del Val, A. and Y. Shoham (1993). Deriving Properties of Belief Update from Theories of Action (II). In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-93)*, Chambery, pp. 732–737.

Gelfond, M., V. Lifschitz, and A. Rabinov (1991). What are the Limitations of the Situation Calculus? In S. Boyer (Ed.), *Automated Reasoning, Essays in Honor of Woody Bledsoe*, pp. 167–181. Kluwer Academic Publishers.

Gustafsson, J. and P. Doherty (1996). Embracing Occlusion in Specifying the Indirect Effects of Actions. In *Proceedings of the 5th Int'l Conference on the Principles of Knowledge Representation in Reasoning (KR'96).*

Lifschitz, V. (1990). Frames in the space of situations. *Artificial Intelligence 46*, 365–375.

Lifschitz, V. (1994). Circumscription. In *Handbook of Logic in AI and Logic Programming*, Volume 3, pp. 298–352. Oxford University Press.

Lin, F. (1995). Embracing Causality in Specifying the Indirect Effects of Actions. In *Proc. International Joint Conference on Artificial Intelligence, Montreal*, pp. 1985–1991. Morgan Kaufmann.

Lin, F. (1998). On the Relationships between Static and Dynamic Causal Rules in the Situation Calculus. In *Proceedings of the AAAI Workshop on Causality.* AAAI. To appear.

Lin, F. and R. Reiter (1994). State Constraints Revisited. *Journal of Logic and Computation 4*(5), 655–678. Special Issue on Actions and Processes.

McCarthy, J. (1986). Applications of Circumscription to Formalizing Commonsense Knowledge. *Artificial Intelligence 28*, 89–116.

McCarthy, J. and P. J. Hayes (1969). Some Philosophical Problems from the Standpoint of Artificial Intelligence. In B. Meltzer and D. Michie (Eds.), *Machine Intelligence* **4**, pp. 463–502. Edinburgh, Scotland: Edinburgh University Press.

Miller, R. and M. Shanahan (1994). Narratives in the Situation Calculus. *The Journal of Logic and Computation 4*(5), 513–530.

Pinto, J. (1994, February). *Temporal Reasoning in the Situation Calculus.* Ph. D. thesis, Department of Computer Science, University of Toronto, Toronto, Ontario, Canada. URL = ftp://ftp.cs.toronto.edu/~cogrob/jpThesis.ps.Z.

Pinto, J. (1998). Occurrences and Narratives as Constraints in the Branching Structure of the Situation Calculus. *Journal of Logic and Computation.* To appear.

Pinto, J. and R. Reiter (1993, June). Temporal Reasoning in Logic Programming: A Case for the Situation Calculus. In D. S. Warren (Ed.), *Proceedings of the Tenth International Conference on Logic Programming*, Budapest, pp. 203–221. The MIT Press.

Reiter, R. (1991). *The Frame Problem in the Situation Calculus: A Simple Solution (sometimes) and a completeness result for goal regression*, pp. 359–380. Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy. San Diego, CA: Academic Press.

Reiter, R. (1993, December). Proving Properties of States in the Situation Calculus. *Artificial Intelligence 64*(2), 337–351.

Reiter, R. (1996, November). Natural Actions, Concurrency and Continuous Time in the Situation Calculus. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference(KR'96)*, Cambridge, Massachussetts, U.S.A. Morgan Kaufmann.

Schubert, L. (1990). Monotonic solution of the frame problem in the situation calculus: an efficient method for worlds with fully specified actions. In H. Kyberg, R. Loui, and G. Carlson (Eds.), *Knowledge Representation and Defeasible Reasoning*, pp. 23–67. Kluwer Academic Press.

Shanahan, M. P. (1997). *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia.* MIT Press.

Thielscher, M. (1997). Ramification and causality. *Artificial Intelligence 89*, 317–364.

Logic Based Modelling of Goal-Directed Behavior

Erik Sandewall
Department of Computer and Information Science
Linköping University, Sweden
E-mail `ejs@ida.liu.se`

http://www.ida.liu.se/ext/etai/received/actions/006/aip.html

Abstract

We address the problem of characterizing goal-directed robotic behavior using a logic of actions and change. Our approach is based on distinguishing two kinds of actions: *procedural* actions which are defined in a mechanistic way, and *goal-directed* actions which are performed through a process involving tries, possibly failures, and corrective action and new tries until the goal has been reached. (The definition of procedural actions may be done external to the logic, for example through differential equations, or through a conventional programming language). For both kinds of actions, the logic expresses explicitly whether the action *succeeds* or *fails*. Each execution of a goal-directed action is also characterized by a number of *breakpoints* where some sub-action has been completed and a new sub-action for getting to the desired goal is selected. The logic is used for characterizing the selection of sub-actions at breakpoints, and the success or failure of the goal-directed action in terms of the success or failure of the sub-actions.

The article describes how goal-directed actions can modelled by an extension of existing results on logics of actions and change.

The article's web page

The web page mentioned at the top of this page will be permanently maintained with information pertaining to the present article. It is used for the annex that will be mentioned below, and we plan to keep it updated with links to additional, relevant information.

1 Topic and approach

The purpose of the present work is to characterize the behavior pattern of *deliberated retry* in logicist terms. Deliberated retry is where an agent pursues goals by trying actions or action sequences that are likely to achieve the goal, and where the agent responds to failure by trying an alternative plan or action sequence. (We identify plans with action sequences). The paper proposes a first-order theory using a narrative timeline approach in which each model is a history of the world where the agent's successive actions exhibit deliberated retry. This work is intended to be used in the design of autonomous agents, and in particular in our WITAS project which is concerned with the control system of an intelligent airborne vehicle (UAV).

A logic of this kind must of course build on existing work in logics of actions and change, but it also imposes some specific requirements on the host logic. In particular, it must allow for the characterization of the success and failure of actions, for external events that influence the execution of an action, and for the description of actions on different levels of resolution. The present section discusses these requirements on the host logic.

1.1 Goals vs high-level actions

We shall consider goals as the end states of particular kinds of actions, namely *goal-directed* actions. Therefore, the question of characterizing goal-directed behavior is reduced to the question of how goal-directed actions are decomposed into procedural ones. By this approach we avoid the need to introduce goals as a separate type of entities, and we open for the possibility (although it will not be used in the present paper) of having goals and actions on several levels.

A goal-directed action is then viewed as an open-ended process: the robot sets out to achieve a certain goal, for example to find and retrieve a particular object, or to obtain more fuel. It chooses certain lower-level actions which are likely to achieve the goal, but these lower actions may succeed or fail; if they fail then the robot diagnoses the error and tries to achieve the same goal in some other way. At the lowest level in this structure one finds the procedural actions, but above them there is a structure of goal-directed actions, possibly on several levels. They are related and held together by decision-like entities and other events, for example decision to try, faults, diagnoses, and decisions about retries. In some cases this structure will be quite simple; in other cases it may be very complex.

The problem addressed here is how such goal-directed behavior can be characterized in a logic of actions and change. We propose an extended action logic whose expressive power is sufficient for this purpose. Concretely, the characteristic property of this logic of action and change is that for a well chosen axiomatization of a scenario, the models represent exactly those histories where the robot exhibits the appropriate, goal-directed behavior. In that sense, the proposed logic is a characterization of goal-directed behavior.

An immediate consequence of this approach is that we need a logic that can represent concurrent actions, since the high-level 'goal' action and the low-level 'process' action are by definition concurrent.

One limitation of the present article is that we only consider the relationship between lower-level and higher-level actions inside the deliberative layer. In the end it will be necessary to deal with influences from external events, the deliberative layer, and concurrent events in an integrated fashion, requiring a closer integration with the earlier work, but we begin here with the simpler case where the latter two aspects are omitted for the time being.

1.2 Defining actions on the level of hybrid processes

Besides the requirement that our logic must be able to express concurrent actions, we also want it to be able to express continuous and hybrid change (hybrid = combination of continuous and discrete). Although it is maybe not strictly necessary, we introduce this requirement for two reasons. First, since the continuous-level description of the world if often needed for defining when actions actually succeed or fail. Secondly, since we intend the logic to be used for autonomous robots where the capability for continuous-level reasoning is needed anyway, so we must make sure that the logic we are using doesnot inherently preclude such applications.

The continuous or hybrid level may be characterized by sets of differential equations which are associated with applicability conditions, so that different equations are stated to apply in different segments of the state space at hand. In earlier papers, we have shown how to import this representation into a logical framework [Sandewall1989a] and how to combine it with specifications of the effects of actions [Sandewall1989b]. More recently, we have also shown how to relate high-level and low-level definitions of actions in such a framework, and in particular how to use the hybrid-level action description for defining success and failure of actions [Sandewall1996, Sandewall1997]. Those papers complement the work reported here.

Concretely speaking, there are quite a number of intervening events that may cause the failure of an action in a physical world. For example, consider an intelligent UAV that is to fly over traffic scenes, and that is required to 'understand' what happens in those scenes, and to take appropriate action towards certain goals, for example for assisting one or more cooperating ground vehicles in their missions. This application involves actions that are performed over periods of time (for example, "follow that car", "accompany the cooperating car to its destination", or "find a place satisfying certain conditions". Reasons why such actions can fail include the actions of the ground vehicles, obstructions to vision or to communication, restrictions on fuel or other resources in the UAV, etc.

The articles that were cited above describe methods for relating the hybrid-level specification of an action to any number of environmental events which may influence or distract it, and for relating the global, precondition/ postcondition type description of the action to the one on the hybrid level.

From the point of view of the hybrid specification of the action, it makes no difference if it is stopped because of an intervention by a person who commands the robot (which counts as an external event as modelled in our earlier work), or because the deliberative layer at some point realized an upcoming danger and decided on an interrupt. Therefore, we can now address the question of modelling high-level behavior that reasons about and uses the success and the failure of subordinate actions.

2 Representing success and failure

A logic for characterizing goal-oriented behavior must be able to express that an action has 'succeeded' or 'failed', since a very important aspect of such behavior is that the agent should try again when one attempt to achieve the given goal has failed. We obtain our logic by extending an existing, narrative time-line logic[1] with a few additional constructs. The present section describes this logic in reasonably precise terms while omitting some of the routine an tedious details.

2.1 Standard Time and Action Logic

The following is the basic notation that we inherit from earlier work. Three primary predicates are used. The predicates H for *Holds* and D for *Do* are defined as follows. $\mathsf{H}(t,p)$ says that the "propositional fluent"

[1]A first-order, multi-sorted logic where time, represented by real numbers, is one of the sorts.

([2]) p holds at time t. In other words, p is reified and $\mathsf{H}(t, p)$ is the same as $p(t)$ in the case where p is atomic. $\mathsf{D}([s, t], a)$ says that the action a is performed over exactly the closed temporal interval $[s, t]$. Open and semiopen intervals are denoted (s, t), $[s, t)$, and $(s, t]$ as usual.

Non-propositional fluents are also admitted, using the notation $\mathsf{H}(t, f : v)$ where $f : v$ is the proposition saying that the fluent f has the value v. Fluent-valued functions are allowed, and one of their uses is to define fluents for properties of objects. Thus $ageof(p)$ may be the fluent for the age of the person p, used as in $\mathsf{H}(1998, ageof(john) : 36)$.

The third predicate, X is pronounced *occludes* and is used for characterizing exceptions from the assumption of continuity of the value of fluents. Continuity includes persistence as a special case, for discrete-valued fluents. $\mathsf{X}(s, f)$ expresses that at time s, the value of the fluent f is not required to be continuous or to persist.

In all cases, s and t are timepoints (usually s for starting time and t for termination time) and a is an action.

In narrative time-line approaches, each model of the axioms characterizes one possible history in the world, not a tree of possible histories. Alternative histories are represented by different models[3]. Therefore, a timepoint t is sufficient for identifying the state of the world at time t in the present model.

Several earlier publications by ourselves and others in our group have used the notations $[t]p$ and $[s, t]a$ for what is here written $\mathsf{H}(t, p)$ and $\mathsf{D}([s, t], a)$, respectively. The change is made in order to emphasize more strongly that we are dealing with a fairly standard first-order logical theory.

2.2 Ontology for invocation and success

Since the performance of a goal-directed action involves trying lower-level actions which may succeed and fail, and to proceed accordingly, we need a notation for dealing with the applicability, success, and failure of actions.

The following notions will be used. To a first approximation, *invocation* of an action causes it to begin its *execution*, which ends with either *success* or *failure*. However, the matter is complicated by the requirement to represent that it is sometimes impossible to execute an action. In our approach, invocation of an action is possible at any time, but the invocation does not necessarily lead to the execution of the action. In particular, it does not if the action is inapplicable by definition (for example, turning on the light in a room where there is no light) or if the action is already executing. The latter condition means that the same action can not execute over two overlapping but non-equal intervals of time.

Once an execution does execute, it must either *succeed* or *fail*. The distinction between success and failure is done on the following pragmatic grounds: planning goal achievement is done using the assumption that actions succeed, and using knowledge about their results when they do succeed. The case where an action fails is dealt with on a case-by-case basis once the failure has occurred.

Each action has a temporal duration, which must be an interval that is greater than a single point except for some specific cases defined below. Note, in particular, that when an action is not applicable, it is considered not to execute; it is not considered to fail instantly. (The reasons for these ontological choices will be briefly explained below).

2.3 Syntax for invocation and success

Two representations will be used for the expression of success, failure, and applicability of actions. In one, we use specially constructed fluents; in the other, variants of the D predicate that distinguish between action success and action failure. The former representation is considered as the basic one, and the latter is introduced as abbreviations or 'macros'.

The following are three functions from actions to propositional fluents:

inv, where $\mathsf{H}(s, inv(a))$ says that the action a is invoked at time s. At all other times, $\mathsf{H}(s, inv(a))$ is false.

app, where $\mathsf{H}(s, app(a))$ says that the action a is applicable at time s.

fail, where $\mathsf{H}(t, fail(a))$ says that the action a terminated with failure at time t. $\mathsf{H}(t, fail(a))$ is false at all times when the action is not executing, or when it is executing but not terminating, or when it is terminating successfully.

In addition, we need one function from propositional fluents (properly speaking, propositional features) to actions:

test, where $test(p)$ or $test(f : v)$ is an action that is

[2]We have previously tried to maintain a terminological distinction between *fluent* as a *function* from timepoints to corresponding values, and a *feature* as a formal object that designates a fluent. With that terminology, the p and f that occur in the second argument of H are features, not fluents. Similarly, the functions *inv*, *app*, and *fail* that will be introduced later in this section, are functions from actions to features. However, since it is so common to use the word 'fluent' both for the function and its designator, we follow that practice here.

[3]However, it also appears that the it is straightforward to generalize the time domain so that it also accounts for the case of branching time.

always applicable, whose duration is always instantaneous (expressed by $\mathsf{D}([s,s], test(p))$), and that satisfies

$$\mathsf{H}(s, fail(test(p))) \leftrightarrow \neg\mathsf{H}(s,p)$$

In other words, $test(p)$ succeeds at time s iff p is true at s.

The following abbreviations are introduced:

$\mathsf{G}(s,a)$ for $\mathsf{H}(s, inv(a))$: the action a is invoked ("go") at time s

$\mathsf{A}(s,a)$ for $\mathsf{H}(s, app(a))$: the action a is applicable at time s

$\mathsf{D}_s([s,t],a)$ for $\mathsf{D}([s,t],a) \wedge \neg\mathsf{H}(t, fail(a))$: the action a is executed successfully over the time interval $[s,t]$; it starts at time s and terminates with success at time t.

$\mathsf{D}_f([s,t],a)$ for $\mathsf{D}([s,t],a) \wedge \mathsf{H}(t, fail(a))$: the action a is executed but fails over the time interval $[s,t]$; it starts at time s and terminates with failure at time t.

$\mathsf{D}_c([s,t],a)$ for $\exists u[\mathsf{D}([s,u],a) \wedge t \leq u]$: the action a is being executed; the execution started at time s and has not been terminated before time t. (It may terminate at t or later).

$\mathsf{D}_v(s,a)$ for $\mathsf{G}(s,a) \wedge (\neg\mathsf{H}(s, app(a)) \vee \exists s' \exists t[\mathsf{D}([s',t],a) \wedge s' < s < t])$: the action a is invoked at time t but it is either not applicable, or already executing at that time. (This is the case where invocation of the action does not initiate an execution).

For both D_s and D_f, s is the time when the action was invoked, and t is the exact time when it concludes with success or failure.

2.4 Axiomatic characterization

The following set of axioms characterizing the obvious properties of these relations is an adaptation of the axioms reported in [Sandewall1996]. The adaptation is because we here introduced *inv*, *app*, and *fail* as the basic notions, whereas previously the relations G, D_s, etc were considered as basic.

S1. If an action is being executed, then it must have been invoked and be applicable and non-executing at invocation time:

$$\mathsf{D}([s,t],a) \rightarrow \mathsf{H}(s, inv(a)) \wedge \neg\mathsf{D}_v(s,a)$$

This implies:

$$\mathsf{D}([s,t],a) \rightarrow \mathsf{H}(s, inv(a)) \wedge \mathsf{H}(s, app(a)) \wedge \\ \neg\exists s' \exists t[\mathsf{D}([s',t],a) \wedge s' < s < t]$$

S2. If an action is invoked, then it is executed from that time on, unless it is inapplicable, already executing, or composite:

$$\mathsf{H}(s, inv(a)) \rightarrow \exists t[s \leq t \wedge \mathsf{D}([s,t],a)] \\ \vee \mathsf{D}_v(s,a) \vee Composite(a)$$

The predicate *Composite* will be introduced in subsection 2.7; for the present context it can be taken as always false. The reason for excluding this case in this axiom is that for composite actions, there are some additional obstacles where the invocation of an action does not result in its execution.

S3. An action can not take place during overlapping intervals:

$$\mathsf{D}([s,t],a) \wedge \mathsf{D}([s',t'],a) \wedge s \leq s' < t \rightarrow s = s' \wedge t = t'$$

S4,S5. Actions of the form $test(p)$ are always applicable, and instantaneous:

$$\mathsf{H}(s, app(test(p)))$$

$$\mathsf{D}([s,t], test(p)) \rightarrow s = t$$

S6. All other actions execute over extended periods of time: never immediately, except for actions of the form $test(p)$:

$$\mathsf{D}([s,t],a) \rightarrow s < t \vee \exists p[a = test(p)]$$

S7. Actions only fail at the end of their execution:

$$\mathsf{H}(t, fail(a)) \rightarrow \exists s[\mathsf{D}([s,t],a)]$$

S8. Definition of success for actions of the form $test(a)$:

$$\mathsf{D}([s,s], test(p)) \rightarrow (\mathsf{H}(s, fail(test(p))) \rightarrow \neg\mathsf{H}(s,p))$$

Several of these axioms capture desirable properties directly. For others, all the consequences are not immediately obvious. One useful consequence is the following theorem, previously mentioned in [Sandewall1996] for a somewhat different axiomatization:

Theorem 1 *In any model for the axiom S3, let $\{[s_i,t_i]\}_i$ be the set of all intervals such that $\mathsf{D}([s_i,t_i],a)$ for a specific action a. Then there is some ordering of these intervals such that $s_i < s_{i+1}$ and $t_i \leq s_{i+1}$ for all i.*

Proof. Suppose the proposition does not hold, and choose an order of the pairs such that $s_i \leq s_{i+1}$, and where each pair only occurs once. Also, choose j so that either $s_j = s_{j+1}$, or $s_j < s_{j+1} < t_j$. If no such j is to be found, then the ordering already satisfies the condition in the proposition.

However, the case $s_j = s_{j+1}, t_j \neq t_{j+1}$ contradicts axiom (S3). The case $s_j < s_{j+1} < t_j$ also contradicts axiom (S3). This concludes the proof. QED

The value of this observation is that through it, it makes sense to use the feature $fail(a)$ for characterizing the success or failure of an action with extended duration. If theorem 1 were not to hold, then it would not be clear from $\mathsf{H}(t, fail(a))$ which invocation the failure referred to. This consideration is also the reason for the choice manifested in axiom S1: if an action a is invoked while it is already in the midst of executing, then it is not represented as "failing", since this would confuse matters with respect to the already executing instance. Instead, we use the convention that it is invoked, possibly applicable, but it does not get to execute from that starting time.

We also obtain at once:

Theorem 2 *In any model for the axioms S1 - S8, if* $\mathsf{D}([s,t],a)$ *and* $\mathsf{H}(u, fail(a))$ *for some* u *in* $(s,t]$, *then* $t = u$. *Conversely, if* $\mathsf{D_s}([s,t],a)$, *then* $\mathsf{H}(u, fail(a))$ *does not hold for any* u *in* $(s,t]$.

Informally, we can think of each model in dynamical terms as a possible history in the world being described, and what this theorem says is that if an action is invoked and begins to execute, then if $\mathsf{H}(u, fail(a))$ becomes true at some timepoint u during the execution, the action halts and ends with failure, and if it is able to proceed until its normal ending without $\mathsf{H}(u, fail(a))$ becoming true at any time, then it ends with success.

Any use of this logic will naturally be concerned with the effects of actions. In the Features and fluents approach and its successors, this is specified using action laws, which in particular make use of the occlusion predicate, and in combination with assumptions of persistence.

2.5 Examples

As an example of the use of this notation, here is the formula stating that a condition φ guarantees that an action always succeeds:

$$\mathsf{H}(s,\varphi) \wedge \mathsf{G}(s,a) \rightarrow \exists t[\mathsf{D_s}([s,t],a)]$$

Ordinary action laws specify the action's effects when it succeeds. They are therefore written as usual and with $\mathsf{D_s}$ on the antecedent side: if preconditions apply and the action is performed successfully, then the postconditions result.

As another simple example, consider the case of actions which are described in terms of a precondition, a prevail condition, and a postcondition, where the postcondition is at the same time the termination condition for the action [Sandewall and Rönnquist1986]. The prevail condition must be satisfied throughout the execution of the action; if it is violated then the action fails. Simple pre/ post/ prevail action definitions can be expressed as follows, if φ_a is the precondition of the action a, ω_a is the postcondition, and ψ_a is the prevail condition:

$$\mathsf{A}(s,a) \leftrightarrow \mathsf{H}(s,\varphi_a)$$
$$\mathsf{D_s}([s,t],a) \rightarrow \mathsf{H}(t,\psi_a \wedge \omega_a)$$
$$\mathsf{A}(s,a) \wedge \mathsf{D_c}([s,t],a) \rightarrow \mathsf{H}([s,t),\psi_a \wedge \neg\omega_a)$$
$$\mathsf{D_c}([s,t],a) \wedge \neg\mathsf{H}(t,\psi_a) \rightarrow \mathsf{D_f}([s,t],a)$$

The traditional case of only pre- and postconditions is easily obtained by selecting ψ_a as tautology.

2.6 Composition operators for propositional fluents

The standard propositional operators such as $\neg$ and $\wedge$ will be used for composing propositional fluents (properly speaking: features). Composition is defined in a Herbrand style, so composite fluents are only equal if they have been equally formed. We have to specify how such composite fluents behave in relation to each of the predicates and functions that can take fluents as arguments:

1. The behavior of composite fluents with respect to the *Holds* predicate is defined by

$$\mathsf{H}(s,p \wedge p') \rightarrow \mathsf{H}(s,p) \wedge \mathsf{H}(s,p')$$

and similarly for the other operators.

2. Composite fluents are always occluded, so that

$$\mathsf{X}(s,p \wedge p')$$

and similarly for the other operators. This means that assumptions of continuity and persistence are only applied to the level of elementary fluents.

3. The action $test(p)$ for composite p must then again be described with respect to how it relates to predicates and functions that take actions as arguments. All that was said about *test* above continues to hold, of course: it is always applicable, its duration is always instantaneous, and it succeeds or fails depending on whether p currently holds or not. No additional axiom is needed or appropriate for the special case where p is composite.

2.7 Action composition operators

The definitions of procedural actions must sometimes be constructed by composition of simpler actions. This calls for the use of operators such as ; for the sequential composition of actions, a conditional operator, and an operator that composes actions representing successive tries. Sequential composition is such that if the first action fails, then the whole action has failed, otherwise it is up to the second action. Successive-try composition, on the other hand, is defined so that if the first action *succeeds*, then the whole actions has succeeded; if the first action fails, then it is up to the second action to succeed or fail.

These action composition operators are best described in terms of their relationships with the derived predicates G, $\mathsf{D_s}$, etc. This is as follows for ;

$$\mathsf{G}(s,a_1;a_2) \rightarrow \mathsf{G}(s,a_1) \wedge$$

$$
\begin{array}{l}
(\mathsf{D}_v(s, a_1) \rightarrow \mathsf{D}_v(s, a_1; a_2)) \wedge \\
(\mathsf{D}_f([s,t], a_1) \rightarrow \mathsf{D}_f([s,t], a_1; a_2)) \wedge \\
(\mathsf{D}_s([s,t], a_1) \rightarrow \mathsf{G}(t, a_2) \wedge \\
\quad (\mathsf{D}_v(t, a_2) \rightarrow \mathsf{D}_f([s,t], a_1; a_2)) \wedge \\
\quad (\mathsf{D}_f([t,u], a_2) \rightarrow \mathsf{D}_f([s,u], a_1; a_2)) \wedge \\
\quad (\mathsf{D}_s([t,u], a_2) \rightarrow \mathsf{D}_s([s,u], a_1; a_2)))
\end{array}
$$

Then, $try(a_1, a_2)$ denotes successive-try composition:

$$
\begin{array}{l}
\mathsf{G}(s, try(a_1, a_2)) \rightarrow \mathsf{G}(s, a_1) \wedge \\
\quad (\mathsf{D}_s([s,t], a_1) \rightarrow \mathsf{D}_s([s,t], try(a_1, a_2)) \wedge \\
\quad (\mathsf{D}_f([s,t], a_1) \rightarrow \mathsf{G}(t, a_2) \wedge \\
\quad\quad (\mathsf{D}_v(t, a_2) \rightarrow \mathsf{D}_f([s,t], try(a_1, a_2)) \wedge \\
\quad\quad (\mathsf{D}_f([t,u], a_2) \rightarrow \mathsf{D}_f([s,u], try(a_1, a_2)) \wedge \\
\quad\quad (\mathsf{D}_s([t,u], a_2) \rightarrow \mathsf{D}_s([s,u], try(a_1, a_2)))) \wedge \\
\quad (\mathsf{D}_v(s, a_1) \rightarrow \mathsf{G}(s, a_2) \wedge \\
\quad\quad (\mathsf{D}_v(s, a_2) \rightarrow \mathsf{D}_v(s, try(a_1, a_2)) \wedge \\
\quad\quad (\mathsf{D}_f([s,t], a_2) \rightarrow \mathsf{D}_f([s,t], try(a_1, a_2)) \wedge \\
\quad\quad (\mathsf{D}_s([s,t], a_2) \rightarrow \mathsf{D}_s([s,t], try(a_1, a_2))))
\end{array}
$$

Notice that if one omits the case of D_v, then the specifications of ; and *try* are symmetrical.

Although these specifications are easy to follow, they are not appropriate as axioms, since the operators G, D_s etc are not in themselves the primary ones. The Annex (see URL at the beginning of the article) contains an axiomatization that relates ; and *try* to *inv*, *app*, *fail*, and *test* and from which the specifications above can be inferred.

The action composition operators represent a kind of "programming language" for procedural actions. Actions which are defined in this way are however still not goal-directed in the sense discussed in the initial section. We shall proceed to goal-directed actions in the next section.

Composition of actions of the form $test(p)$ can be reduced to propositional combinations, for example

$$\mathsf{D}_s([s,s], test(p); test(q)) \leftrightarrow$$

$$\mathsf{D}_s([s,s], test(p)) \wedge \mathsf{D}_s([s,s], test(q))$$

We choose to *define* composition of such actions in terms of equality:

$$test(p); test(q) = test(p \wedge q)$$

$$try(test(p), test(q)) = test(p \vee q)$$

The definitions of "if" and "while" are obtained in similar manner, and are detailed in the Annex. (Note that "if p then a else b" can not be expressed using the operators defined so far).

The predicate *Composite*(a) is defined so that it is true for actions a that are formed using the functions ;, *try*, etc., except for the ones where all components are of the form $test(p)$ so that they are equal to non-composite actions by what has just been said. It is straight-forward to write out the axioms for the definition of *Composite*.

3 Deliberated retry

We proceed now to the phenomenon of *deliberated retry*, which is characteristic of high-level actions: if something goes wrong, then try again, but before you do that, consider carefully what different options are available. The weighing of possible alternatives is what differentiates deliberated retry from the preprogrammed successive-try composition defined above.

3.1 Ontology

The goal-directed behavior that we wish to characterize in the logic is as follows. At each point in time, the robot is engaged in no, one, or more *processes*, each of which is an instance of goal-directed behavior. Each process goes on for an interval of time, then it ends and once ended, the same process can not restart. At each point in time within its duration, the process is carrying out a *plan*, which is an action and in the general case a composite action, formed using the action composition operators that were defined in section 2.7. Each constituent action may succeed or fail, which also defines the success or failure of the plan. If the current plan fails, then another plan is found, if possible, for achieving the current goal. If the current plan succeeds, then the current process succeeds. Therefore, there is an implicit assumption that the choice of plans and the definition of success of plans is such that the success of the plan guarantees that the goal has been achieved. If no applicable plan exists, then the process fails.

The robot agent controls this process in the following ways:

- The agent invokes a goal-directed process by stating the formula $\mathsf{H}(s, inv(g))$.
- The agent discontinues an on-going process by stating the formula $\mathsf{H}(t, fail(g))$.
- The agent discontinues an on-going action within a process by stating the formula $\mathsf{H}(t, fail(a))$.
- The agent selects the new plan to be used within a process when a current plan has failed. This is done by stating the formula $\mathsf{H}(s, inv(a))$ where a is often a composite action.

Notice that in all cases, the agent exercises its control by making statements of the form $\mathsf{H}(t, p)$ where p has the property of being true at singular points in time, and false everywhere else. It is appropriate to think about such fluents as *signals*.

It is assumed that the agent exercises these capabilities correctly, so that e.g. it does not discontinue processes that are not in course, it only selects plans that are guaranteed to achieve the goal if successful, etc.

For simplicity, we assume here that each process is linear in the sense that it is not able to spawn other, concurrent processes. This is in line with the fact that no action composition operator for concurrent execution was introduced in subsection 2.7.

Possible concurrent occurrences of the same action are assumed to behave in line with the formalization in the preceding sections. This means that if several concurrent processes request the same action a to be invoked at the same time, then this can be done, but only one instance of the action is invoked. If that instance fails, then the failure affects all the invoking processes On the other hand, if a process invokes an action a at a time where the same action is already in course in another, concurrent process, then the new invocation of the action a falters (no new execution is initiated).

3.2 Additional formalism

Previous authors have sometimes used a modal operator for specifying goals. Here, we manage with a simpler approach, essentially because we have not set out to characterize goals, but only to characterize goal-directed behavior.

The basic idea has already been mentioned: We distinguish actions of several levels. The lowest level of action is the procedural one, the one which is implemented as a program or other routine behavior. Working towards a goal is represented as a higher level action, often realized by performing several low-level actions in succession.

Thanks to this approach, we only need to make some simple additions to the background formalism that was briefly reviewed above. We introduce a few specialized predicates, besides the general-purpose predicates H, D, and X. The new predicates are:

- $Option(s', g, s, a)$ which says that while performing the high-level and goal-directed action g that was invoked at time s', at time s the goal has not yet been achieved, and the world is in a state where the action a is executable, and where its successful execution will achieve the goal.
- $Realize(s, g, a)$ which says that at time s, the standard way of performing the (high-level) action g is to initiate the (low-level) action a.

In implementation terms, the predicate *Option* encapsulates the system's replanner: if the high-level action g was invoked at time s', and one attempt to achieve the goal has just failed at the present time s, then $Option(s', g, s, a)$ shall be true for exactly those (composite) actions a that will take the robot the remaining distance to the given goal Similarly, $Realize(s, g, a)$ shall be true for at most one a for given s and g, namely for that a which is the standard way of achieving goal g from the state of the world at time s.

What if a proposed action or plan a is nondeterministic and *possibly* achieves the goal, but is not guaranteed to do so? In this case, one can always use $a; test(p)$ as the last argument of *Option*, where p is the desired goal condition. The action $a; test(p)$ will execute a, and if it succeeds then it tests whether p is true and if so it succeeds, otherwise the whole action fails.

It is assumed that these relations satisfy

$$Option(s', g, s, a) \rightarrow \mathsf{H}(s, app(a))$$

$$Realize(s, g, a) \rightarrow Option(s, g, s, a)$$

Furthermore, if $achieve(p)$ is the generic goal-directed action having the property

$$\mathsf{D}_s([s, t], achieve(p)) \rightarrow \mathsf{H}(t, p)$$

that is, the action succeeds when p has been achieved, then

$$Option(s', achieve(p), s, a)$$

should be true for every a satisfying

$$Option(s', achieve(p), s, a) \wedge \mathsf{D}_s([s, t], a) \models \mathsf{H}(t, p)$$

in the presence of the other axioms as specified in section 4 below. The variable symbol a will be used for low-level actions, and g for high-level actions, and when explicitly or implicitly quantified they only range over those respective subtypes. At present, these subtypes are kept distinct, but we foresee a generalization where actions may be decomposed successively through several levels. The predicate D and the functions on actions (*inv*, *app*, *succ*) apply equally to high-level and low-level actions.

Finally, we need one more variant of the D predicate, this one defined directly and not as an abbreviation. The formula $\mathsf{D}_b([s, t], g)$ will express that the goal-directed action g was invoked at time s, that at time t it has not yet succeeded, and that time t is a *breakpoint* in the sense that one of the attempts to achieve the goal has just failed, and the robot is considering what to do next. For given s and g where $\mathsf{G}(s, g)$ is true, $\mathsf{D}_b([s, t], g)$ will be true for all t which are breakpoints during the process of trying to reach the goal specified by g, and for no other t.

3.3 Axiomatization for the sequential case

The case of several concurrent processes has the particular complication that a proposed invocation of a step in a plan may falter because the same action is presently in the midst of executing. One must then decide whether to replan, or to wait until the action becomes available and then perform it, or whether possibly it is sufficient to use the state of the world at the end of the present execution of the action. This choice is problem-dependent, which contributes to the complexity of concurrency in this context.

In the purely sequential case these problems do not occur, and we shall therefore treat them first; in this paper we limit our attention to them. The ontology descried above is characterzied by the following axioms, in addition to those defined in earlier sections.

G1. $\mathsf{G}(s,g) \rightarrow \mathsf{D_b}([s,s],g)$

G2. $\mathsf{D_b}([s,t],g) \wedge \forall a[\neg Option(s,g,t,a)] \rightarrow$
$(s = t \rightarrow \mathsf{D_v}(s,g)) \wedge$
$(s < t \rightarrow \mathsf{D_f}([s,t],g))$

G3. $\mathsf{D_b}([s,t],g) \wedge Option(s,g,t,a) \rightarrow$
$\mathsf{D_f}([s,t],g) \vee$
$\exists a'[\mathsf{G}(t,a') \wedge Option(s,g,t,a') \wedge$
$(\mathsf{D_s}([t,t'],a') \rightarrow \mathsf{D_s}([s,t'],g)) \wedge$
$(\mathsf{D_f}([t,t'],a') \rightarrow \mathsf{D_b}([s,t'],g))]$

These axioms are organized as a kind of "engine" for doing the goal-directed behavior, based on the notion of breakpoints, that is, points where a plan has failed and replanning has to take place. Axiom G1 says that immediately when a goal-directed action has been invoked, you are at a breakpoint for that action. Axiom G2 says that if you are at a breakpoint and no plan is available, then the process fails. Axiom G3 says that if you are at a breakpoint and some plan *is* available, then *some* plan (not necessarily the one mentioned in the antecedent) is invoked. It further says that if the selected plan succeeds, then the goal-directed action succeeds, and if the invoked plan fails, then the process is at a new breakpoint where replanning has to take place again.

Axiom G3 has an additional literal for the possiblity that the goal-directed action fails. This literal is intended to cover the case that the robot decides to discontinue the process exactly when it is at a breakpoint. If the robot discontinues the goal while being within the execution of a plan, we need instead the following axiom:

G4. $\mathsf{D_f}([s,t'],g) \wedge Option(s,g,t,a) \rightarrow$
$(\mathsf{D_c}([t,t'],a) \rightarrow \mathsf{D_f}([t,t'],a))$

Note that this means that a success of a low level action can be redefined as failure by a high level action. On the other hand, if the robot chooses to fail the ongoing low-level action, the currently executing plan, or some segment of the currently executing plan, then the already defined axioms take care of it correctly.

3.4 Minimization of actions

The axioms that were specified in the previous section defined when certain actions and events must take place, including both the invocation and the failure of actions. They do not restrict the actions and events to the minimally necessary actions or the only motivated events, but presumably such minimization is intended. It would be contrary to the concept of goal-directed behavior to see the occurrence of unmotivated actions, and it would not make ontological sense to see the failure condition for actions trigger at arbitrary times and without reasons.

As usual, there are two ways of eliminating models containing unintended actions and events: by introducing additional axioms, or by an explicit minimization policy on models. In the concurrent case, it turns out to be quite difficult to eliminate all redundant actions by explicit axioms. One can get some of the way. For example, it is straightforward to write an axiom saying that every invocation of an action must be obtained from a *Option* statement. It is also not very difficult to write an axiom saying that if several alternative plans are proposed in the same case of *Option*, only one of them will be chosen. This appears to be sufficient for the non-concurrent case.

Suppose, however, that we have two concurrent processes which happen to have concurrent breakpoints; one of them considers plans a and b; the other one considers plans a and c. We would then accept that only plan a is selected, and possibly that plans b and c are selected, but certainly not that a and b are selected since a can do the job alone. To make matters worse, suppose we have three concurrent processes with shared breakpoint; one of them considers a and b, one considers b and c, and one considers a and c.

An axiomatization that deals correctly with these obstruse cases is likely to become quite complex, and unfortunately the complexity will largely be due to very odd cases. At the same time, it seems that the model-preferential specification can be made quite simple and concise: minimize $\{inv(a)\}_a$ chronologically.

A possible objection to such a principle might be that chronological minimization of effort may be very short-sighted, and sometimes it is important to look ahead and trade off current work against future comfort. However, this misses the point, since such tradeoffs can be done as appropriate in the replanning process and in the choice between alternative plans. The point made here is that *once the plans have been selected* in the participating processes, one only invokes a minimal set of actions at each point in time.

3.5 Characterizing the action processes

A complete cognitive robotics system needs to reason about the effects of actions, so it needs access to action laws (sometimes called action effect laws). In the context of a success/ failure distinction for actions, there is also a need for rules that specify the conditions under which actions may fail or are bound to fail.

None of these considerations have been made in the present article, because we shall use an approach where the treatment of those other aspects are dealt with separately and in a modular way. The present treatment can be limited to the question of the goal-directed behavior as such.

In particular, a software module that makes interpretive use of the present set of axioms will output invocations of actions within plans; it will require several kinds of inputs, including the information about the success or failure of actions and plans, and information about the "moves" of the robot agent as defined in section 3.1. All of this information can be communicated as simple logic formulae.

Notice, in particular, that the module being described here does not need to be involved at the beginning and end of every elementary action in a plan; it is sufficient for it to be involved at the breakpoint where replanning takes place. Notice also that the replanning process itself is encapsulated in the specification of *Option*, which means that it is held open whether replanning is to be done by an inference engine or by some other process, for example an algorithmic process or lookup in a plan library. The only important thing from our point of view is that the fourth argument of the predicate *Option* can be determined when the first three predicates are given, together with the relation H, and that the relation *Option* satisfies the criterium for producing correct plans. Naturally, if *Option* is implemented by an inference based method such as deduction or abduction, then it must make use of action laws expressed in logic, as usual.

In summary, we have now showed how the invocation of a goal-directed action can invoke one or more procedural actions, and how the proper reactions to the success or failure of the latter can be specified in logic. The key notion in this formalisation is that it does not explicitly prescribe *the* next action to be taken in a particular failure situation; the forward deductive machinery is able to derive a number of candidates. The axioms in section 3.3 assure that exactly one of those will be chosen, assuming of course that at least some action is implied to be considered and that there is no overriding command.

The success and failure of the procedural actions is in turn defined on the level of continuous or hybrid description. The previous article [Sandewall1996] addressed how to establish the deliberative-level description of actions, including both the success case and the failure case, as logical consequences of the hybrid-level description.

4 Entailment methods

The previous sections have described a logical machinery for goaldirectedness that requires the use of the following sets of axioms:

- The set **S**, consisting of the axioms S1 – S8 in section 2.4, together with axioms characterizing composite properties (section 2.6) and composite actions (section 2.7 and annex).
- The set **G**, consisting of the axioms G1 – G4 in section 3.3, together with the two axioms in section 3.2.
- A set **R** of behavior rules, consisting of axioms using the predicates *Option* and *Realize* for specifying concrete behaviors.

However, it would not make sense to use them on a stand alone basis. They are intended to define goal-directed behavior in a deliberative context, where the following knowledge sources exist as well:

- A set **E** of action laws specifying the effects of actions when they succeed
- A set **A** of applicability laws, specifying when actions are applicable
- A flow **O** of observations, providing specific facts at specific points in time.
- A flow **D** of decisions by the robotic agent of the kinds specified in section 3.1.

If conventional logic were to be used, then it would be a trivial matter to combine these knowledge sources: one would merely take the union set of all the axioms, and use them for the deductive machinery. In the present setting, however, the matter is more complicated since nonmonotonicity is involved. In particular, it is well known that the action laws **E** need to be used in a nonmonotonic context, if they are written in a reasonable way. We have also observed in an earlier paper [Sandewall1996] that the proper treatment of action failure as a non-standard way of terminating actions requires the use of another kind of nonmonotonicity, and we have observed in subsection 3.4 of this article that minimization of actions calls for yet another kind of nonmonotonicity, at least when concurrency is involved. The question of how to combine the above mentioned knowledge sources is therefore not at all obvious. The present section will provide an answer to this important problem.

4.1 A model example

The general formulation of the problem at hand is the following: *given two or more logical knowledge sources, where each of them specifies some aspect of the dynamic behavior of a system, and where these aspects are interdependent so that the changes imposed by one knowledge source influences the continued development described by the other(s), how are those knowledge sources to be combined in the framework of nonmonotonic logics?* In order to address this question, we first describe a very simple case where the approach can be brought out clearly.

Consider therefore a system where there are two multi-valued fluents a and b for discrete time, and two knowledge sources A and B. A specifies the value of a at the next time-step depending on the values of a and b at the previous time-step, and B specifies the new value for b in the same way. Then, set up the logic so that each interpretation is a mapping from timepoints to corresponding values for a and b. In other words, each interpretation is a possible history of the world. Let $M(A)$ be the set of all interpretations *for arbitrary assignments to b*, and where each interpretation specifies the successive values for a according to its previous value and the value at hand for b. Let $M(B)$ be similar for the fluent b. Obviously, $M(A) \cap M(B)$ is the set of all histories of the world that develop according to the joint information of the two knowledge sources.

Suppose further that A and B are such that they need to be used in the context of a nonmonotonic logic. For example, A may be characterized by inertia or persistence, so that it is a set of rules specifying when the value of the fluent changes; there is a background assumption that if A does not specify any change, then the fluent stays constant from one timepoint to the next. It is well known that in this case, $M(A)$ can be conveniently written e.g. as $Min(<_a, Mod(A))$ where $Mod(A)$ is the set of classical models of A, $<_a$ is a preference relation on models which prefers inertia in the a component between models having the same b component up to the timepoint of comparison, and Min is an operator reducing a set of models to the subset consisting of those members that are minimal with respect to the ordering in the first argument.

Suppose similarly that the knowledge source B has been written using the assumption of a normal value, so that the value of b at any time shall be the normal value unless a rule in B implies otherwise. This case can be dealt with by an approach similar to the one for A, except that another preference relation must be used.

It is now straightforward to see that the two knowledge sources, each having its own nonmonotonic entailment method, can be combined and that the set of selected models for the combination of A and B ought to be

$$Min(<_a, Mod(A)) \cap Min(<_b, Mod(B))$$

With this insight, we can return to the case at hand.

4.2 Entailment method for goal-directed behavior: simple case

We restrict our attention to the case where the agent only invokes goal-directed actions. It does not make an explicit choice between options; that choice is modelled as random (meaning that all choices are obtained as models), and the agent also does not fail actions on any level. Action failure is obtained as observations, that is, from the world at hand. Generalization to the case of more complex agent interactions appears to be fairly straight-forward, and is planned to follow in a later contribution.

Interpretations are constructed as sixtuples $\langle \mathsf{H}, \mathsf{D}, \mathsf{X}, \mathsf{D_b}, Option, Realize \rangle$ in the obvious fashion. Furthermore, in each such interpretation, the H component is partitioned into four parts,

$$\mathsf{H} = \mathsf{H}_{ord} \cup \mathsf{H}_{inv} \cup \mathsf{H}_{fail} \cup \mathsf{H}_{app}$$

where H_{inv} contains value assignments for fluents of the form $inv(a)$, similarly for H_{fail} and H_{app}, and H_{ord} contains value assignments for all other fluents. In line with the approach of the previous subsection, we proceed as follows:

- One set of models is constructed by allowing H, X, and $\mathsf{D_b}$ to vary freely, except H_{app}, and using the axiom sets **S** (characterizing invocation of actions) and **A** (applicability of actions) for properly constraining H_{app} and D.
- Another set of models is constructed by allowing H_{inv}, H_{app}, D, and $\mathsf{D_b}$ to vary freely and using the axiom set **E** (action laws) for obtaining proper execution of actions and proper effects of actions in all different cases of invocation that may arise. This will constrain X, H_{ord}, and H_{fail}.
- Another set of models is constructed allowing free variation of all components except H_{inv} and $\mathsf{D_b}$, and using the knowledge sources **D**, **R**, and **G** in order to constrain H_{inv}, $\mathsf{D_b}$, *Option*, and *Realize*.
- A final set of models is constructed as the classical model set for the knowledge source **O**, that is, the observations. It will only constrain H_{ord}.

The intersection of the model sets obtained in these four ways are clearly the desired ones, provided that each of the participating sets is selected correctly. For the first case, this is straight-forward. The second case is one which has already been studied extensively, and a catalogue of different entailment methods and their respective properties has been published in [Sandewall1994]. The third case is the one that needs to be further considered here. The fourth case requires no minimization of models, and is in that sense trivial.

But considered in this light, the solution to the selection of model set with respect to H_{inv} is also straight-forward: it must be merely a question of minimizing H_{inv} chronologically. In other words, at each point in time, only those actions are invoked whose invocation is necessary due to the given axioms and the situation at hand. Certainly there may be more than one choice of action to be invoked, and this is represented by obtaining several models.

The chronological minimization of H_{inv} at a timepoint t is of course done separately for different histories of H_{ord}, $\mathsf{D_b}$, etc. up to that time. It is intended that the proposed axioms shall characterize exactly the intended occurrences of D_b for given H etc, so no minimization of D_b ought to be required. The use of nochange axioms instead of chronological minimization seems to be straightforward for the non-concurrent case but problematic for the concurrent case, for the reasons that were discussed in subsection 3.4.

5 Discussion

5.1 Limitations of these results

The present treatment is incomplete in a number of ways, in particular:

- We have not yet treated the case of more general agent behavior, where the agent may discontinue goals and specific actions being executed towards the goals.
- We have not yet given complete treatment to the case of concurrent actions, although the formalism allows it and some parts of the present material also allows for that case.
- One particular aspect of the question of concurrent actions is that some of the axioms specified above do not completely characterize what must happen when the goal-directed process proposes to invoke an action that is already in the midst of executing in service of another goal.
- The formulation of a plausible underlying semantics, and a validation of the present approach with respect to that semantics has not yet been undertaken. This also means that the axiomatization proposed here is preliminary and is to be taken as a first proposal which needs to be subjected to additional "debugging".

5.2 Conclusion

We have described a way of characterizing goal-directed robotic behavior by moderate extensions of a current logic of actions and change. In particular, our logical system is able to characterize the sequencing and the choice of successive sub-actions which are performed in order to achieve a given goal, and to infer the success or the failure of the overall goal from the success or failure of the sub-actions. We have also proposed a nonmonotonic entailment method for using the axiomatisation of the present article in conjunction with other, related knowledge sources.

We consider that work on these problems is a necessary step on the way to designing intelligent robotic agents what can be analyzed formally and for which one can prove some of their properties in a systematic fashion.

6 Related work

The material presented in section 2 (D_s and D_f predicates and the definition of composite actions in that context) has been presented in an earlier paper by ourselves [Sandewall1996]. The same holds for the general approach to logics with explicit time, which has been systematically described in [Sandewall1994]. The approach to represent actions with extended duration by an instantaneous invocation, an instantaneous termination, and possible some additional, instantaneous events within the course of the action, is also used in the "new situation calculus" work of Levesque, Reiter et al [Levesque *et al.*1997].

A recent workshop article by De Giacomo, Reiter, and Soutchanski [Giacomo *et al.*1998] describes an approach to execution monitoring using the framework of the previous article. The purpose of their system is to take corrective action after the failure of one step in a composite action (a "plan"), which is of course almost identically the same topic as in our present work. By comparison, their approach appears to be more computation-oriented than ours: they describe recovery in procedural terms, and they do not define the required entailment method. Apart from these differences of emphasis, there are many similarities between their approach and ours.

The work by Kabanza et al [Kabanza *et al.*1997] has also partly similar goals to the present ones, but differs in the sense that they study safety and liveness constraints, in the tradition of the theory of real-time systems. This leads them to an approach based on modal-temporal logic which is considerably more complex than the one proposed here. On the other hand, they only consider the world in terms of discrete state-transition functions, and therefore they do not have the grounding of actions (including action failure) in terms of physical models. This connection is an important aspect of our approach.

The contributions of Levesque, Reiter, et al and of Kabanza are relatively recent ones. The topic addressed here is also related to two strands of earlier research. One is the work on architectures for rational agents, in particular those contributions which attempted to characterize the architecture in precise and formal ways. Rao and Georgeff [Rao and Georgeff1992] have developed an abstract architecture for rational agents based on the concepts of belief, desire, and intention. Their architecture is defined in terms of an abstract interpreter and a formal language that is used for expressing beliefs, intentions, etc. The language includes a number of modal operators which lack a counterpart in our system, such as a belief operator and a concept of "inevitable". It also uses some constructs that we have inherited, in particular the distinction between success and failure of an action. However, their system does not apply the concept of success and failure to the achievement of a goal, and in general it lacks the additional expressiveness that is obtained by treating goals as goal-directed actions.

One important success criterion for a logicist approach to agent behavior is that it ought to be able to characterize those behaviors that have been implemented and found useful in practical agent architectures. It is therefore no coincidence that the ontology and the agent behaviors in our approach shows many similarities with the one used in practical software architectures for autonomous agents, such as the RAPs (Reactive Action Packages) approach of Firby et al [Firby *et al.*1995]. Tate's O-Plan2 system [Tate1994] is an extensive architecture for plan-guided systems, whose behavior is of course considerably more complex than has been modelled in our approach so far. O-Plan2 is only described in terms of its computational processes.

The other relevant strand of earlier work is in the area of goal-directed and plan-based multi-agent systems. In this case, the focus is on the modalities of knowledge and belief, and on communication and knowledge acquisition actions. Physical actions such as the robotic actions that we have discussed here are barely present or not at all. In particular, Shoham [Shoham1993] has introduced a logical system for agent-oriented programming, and Levesque et al [Lespérance and others1995, Levesque1996] introduce an alternative, logical approach to agent programming. These approaches are more or less complementary to ours in the sense that a complete system should include both modal constructs (which they have and we don't) and actions in the physical world as well as a capability for reasoning about the achievement or non-achievement of goals.

A more extensive discussion of related work can be found via the article's web page, which was referenced in the head of the article.

Acknowledgements

This research was conducted within the Wallenberg Laboratory for Information Technology and Autonomous Systems (WITAS), which is supported by the Wallenberg Foundation.

References

[Firby *et al.*1995] R. James Firby, Roger E. Kahn, Peter N. Prokopowicz, and Michael J. Swain. An architecture for vision and action. In *International Joint Conference on Artificial Intelligence*, pages 72–79, 1995.

[Giacomo *et al.*1998] Giuseppe De Giacomo, Ray Reiter, and Mikhail Soutchanski. Execution monitoring of high-level programs. In *Working Papers of Common Sense '98*, pages 277–297, 1998.

[Kabanza *et al.*1997] F. Kabanza, M. Barbeau, and R. St-Denis. Planning control rules for reactive agents. *Artificial Intelligence*, 95(1):67–113, 1997.

[Lespérance and others1995] Yves Lespérance et al. Foundations of a logical approach to agent programming. In *Proc. IJCAI 95 Workshop on Agent Theories, Architectures, and Languages*, 1995.

[Levesque *et al.*1997] Hector J. Levesque, Raymond Reiter, Yves Lesérance, Fangzhen Lin, and Richard B. Scherl. Golog: A logic programming language for dynamic domains. *jlp*, 31(1-3):59–84, April-June 1997.

[Levesque1996] Hector J. Levesque. What is planning in the presence of sensing? In *National Conference on Artificial Intelligence*, 1996.

[Rao and Georgeff1992] Anand S. Rao and Michael P. Georgeff. An abstract architecture for rational agents. In *International Conference on Knowledge Representation and Reasoning*, pages 439–449, 1992.

[Sandewall and Rönnquist1986] Erik Sandewall and Ralph Rönnquist. A representation of action structures. In *National Conference on Artificial Intelligence*, pages 89–97, 1986.

[Sandewall1989a] Erik Sandewall. Combining logic and differential equations for describing real-world systems. In *Proc. International Conference on Knowledge Representation, Toronto, Canada*, 1989.

[Sandewall1989b] Erik Sandewall. Filter preferential entailment for the logic of action in almost continuous worlds. In *International Joint Conference on Artificial Intelligence*, pages 894–899, 1989.

[Sandewall1994] Erik Sandewall. *Features and Fluents. The Representation of Knowledge about Dynamical Systems. Volume I.* Oxford University Press, 1994.

[Sandewall1996] Erik Sandewall. Towards the validation of high-level action descriptions from their low-level definitions. *Artificial Intelligence Communications*, December 1996. Also Linköping University Electronic Press, http://www.ep.liu.se/cis/1996/004/.

[Sandewall1997] Erik Sandewall. Relating high-level and low-level action descriptions in a logic of actions and change. In Oded Maler, editor, *Hybrid and Real-Time Systems*, pages 3–17. Springer Verlag, 1997.

[Shoham1993] Yoav Shoham. Agent-oriented programming. *Artificial Intelligence*, 60(1):51–92, 1993.

[Tate1994] Austin Tate. The emergence of standard planning and scheduling system components. In Christer Bäckström and Erik Sandewall, editors, *Current Trends in AI Planning*. IOS Press, 1994.

$\mathcal{AOL}$: a logic of acting, sensing, knowing, and only knowing

Gerhard Lakemeyer
Department of Computer Science
Aachen University of Technology
D-52056 Aachen
Germany
gerhard@cs.rwth-aachen.de

Hector J. Levesque
Department of Computer Science
University of Toronto
Toronto, Ontario
Canada M5S 3A6
hector@cs.toronto.edu

Abstract

This work is motivated by the existence of two useful but quite different knowledge representation formalisms, the situation calculus due to McCarthy, and the logic $\mathcal{OL}$ of only knowing due to Levesque. In this paper, we propose the logic $\mathcal{AOL}$, which combines both approaches in a clean and natural way. We present a semantics for $\mathcal{AOL}$ which generalizes the semantics of $\mathcal{OL}$ to account for actions, and a sound and complete set of axioms for $\mathcal{AOL}$ which generalizes the Lin and Reiter foundational axioms of the situation calculus to account for only knowing. The logic is compatible with earlier work on knowledge and action in that the solution to the frame problem for knowledge proposed by Scherl and Levesque becomes now a theorem of $\mathcal{AOL}$. We also demonstrate that the logic avoids certain anomalies present in related work by Lakemeyer. Finally we provide a mapping from $\mathcal{OL}$ into $\mathcal{AOL}$ such that a sentence of $\mathcal{OL}$ is valid iff its mapping is a theorem of $\mathcal{AOL}$, thus providing, for the first time, a complete axiomatic characterization of $\mathcal{OL}$.

1 Introduction

This work is motivated by the existence of two useful but quite different knowledge representation formalisms:

- the *situation calculus* [McC63] is a dialect of first order logic for representing and reasoning about the preconditions and effects of actions. A recent second-order refinement explored by Lin and Reiter [LR94] has been shown to be useful for high-level robot and agent control [LRL97], exploiting a simple solution to the frame problem presented in [Rei91].
- the logic of *only knowing* $\mathcal{OL}$ [Lev90] is a quantified modal logic for representing and reasoning about the *de dicto* and *de re* knowledge[1] of an agent, and all that that agent knows.[2] This language has been found useful for capturing autoepistemic reasoning [Moo85b] within a purely monotonic logic, as well as certain forms of relevance [Lak95].

In this paper, we propose the logic $\mathcal{AOL}$, which includes in a clean and natural way both the situation calculus noted above and an embedding of the logic of only knowing. The amalgamation is carried out at both the semantic and the syntactic levels. While the semantics naturally extends the model theory of $\mathcal{OL}$ to account for actions, the axioms, which are sound and complete for the semantics, can be seen as a version of the foundational axioms of the situation calculus proposed in [LR94], generalized to deal with the much richer ontology required for only knowing.

The motivation for the amalgamation is perhaps best seen in the following example. Suppose we have a robot that knows nothing about the initial state of the environment, but that there is a sensing action, reading a sonar, which tells the robot when it is getting close to a wall. Then we would like to prove the following:

1. in the situation that results from reading the sonar, the robot knows whether the wall is close;
2. assuming the robot knows the sonar is working, it also knows in the initial state that it will know whether the wall is close after checking its sonar;
3. suppose the robot checks its sonar and discovers that the wall is not close. If it now moves towards the wall, it no longer knows whether or not the wall is close; if

[1] *De dicto* and *de re* knowledge refers to the distinction between *knowing that* and *knowing who* [Kap71].

[2] Actually, it is belief that is dealt with in $\mathcal{OL}$, but we will use the two terms interchangeably, unless noted otherwise.

it moves away from the wall instead, it continues to know whether or not it is close.

These are all simple and reasonable properties involving knowledge and action. Observe that to get them right, it is necessary to reason about the effects of sensing actions, knowledge about knowledge and action, and all that is known. The latter is necessary, in particular, if an agent wants to reason about its own ignorance without having to be told explicitly what it does not know. Furthermore, knowledge about ignorance plays an important role in guiding an agent's actions such as deciding whether it is necessary to use a sensor.

Ours is certainly not the first approach combining knowledge and action (see, for example, [Moo85a, SL93, Lak96]). In fact, $\mathcal{AOL}$ is compatible with this line of work in that the solution to the frame problem for knowledge proposed by Scherl and Levesque [SL93], which builds on [Moo85a], becomes a logical consequence of the axioms. So far, only Lakemeyer [Lak96] has proposed an amalgamation of only knowing and action in a quantified logic.[3] However, he provides only a semantics, but no axioms. In this paper we also point to certain anomalies in his logic and demonstrate how they are avoided in $\mathcal{AOL}$. Finally we provide a mapping from $\mathcal{OL}$ into $\mathcal{AOL}$ such that a sentence in $\mathcal{OL}$ is valid iff its mapping into $\mathcal{AOL}$ is a logical consequence of the axioms. This provides, for the first time, a complete axiomatic characterization of $\mathcal{OL}$.

The rest of the paper is organized as follows. In Sections 2 and 3, we briefly review the situation calculus and the logic $\mathcal{OL}$, respectively. In Section 4, we present the new logic $\mathcal{AOL}$, both semantically and axiomatically. The properties of knowledge and action are studied in more detail in Section 5. Section 6 formalizes the robot example above. In Section 7, we show an embedding of $\mathcal{OL}$ into $\mathcal{AOL}$. In Section 8, we compare our approach with the amalgamation presented in [Lak96]. Section 9 presents a brief summary and suggests areas of future work.

2 Situation Calculus

One increasingly popular language for representing and reasoning about the preconditions and effects of actions is the situation calculus [McC63]. We will only go over the language briefly here noting the following features: all terms in the language are one of three sorts, ordinary objects, actions or situations; 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; similarly, functions varying across situations are called functional fluents and are denoted analogously; finally, there is a special predicate $Poss(a, s)$ used to state that action a is executable in situation s.

Within this language, we can formulate theories which describe how the world changes as the result of the available actions. One possibility is a *basic action theory* of the following form [Rei91]:

- 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 the so-called effect axioms, but also provide a solution to the frame problem [Rei91].
- Domain closure and unique names axioms for the primitive actions.
- A collection of foundational, domain independent axioms.

In [LR94] the following foundational axioms are considered:[4]

1. $\forall s \forall a. S_0 \neq do(a, s)$.
2. $\forall a_1, a_2, s_1, s_2.\, do(a_1, s_1) = do(a_2, s_2) \supset$
   $(a_1 = a_2 \wedge s_1 = s_2)$.
3. $\forall P.\, P(S_0) \wedge [\forall s \forall a.(P(s) \supset P(do(a, s)))] \supset$
   $\forall s P(s)$.
4. $\forall s.\, \neg(s < S_0)$.
5. $\forall s, s', a.\, (s < do(a, s') \equiv (Poss(a, s') \wedge s \leq s'))$,
   where $s \leq s'$ is an abbreviation for $s < s' \vee s = s'$.

The first three axioms serve to characterize the space of all situations, making it isomorphic to the set of ground terms of the form $do(a_1, \cdots, do(a_n, S_0)\cdots)$. The third of these is a second-order induction axiom that ensures that there are no situations other than those accessible using *do* from S_0. The final two axioms serve to characterize a $<$ relation between situations. Later, we will be introducing abbreviations of various sorts into our representation language,

[3]In the context of modeling belief revision, [dVS94] provide axioms for only knowing in the situation calculus, but these are limited to the propositional case.

[4]In addition to the standard axioms of equality

and we can in fact do so here, defining the $<$ relation as an abbreviation for a certain second-order formula:

$$s < s' \doteq \forall P[\ldots \supset P(s, s')]$$

where the ellipsis stands for the conjunction of

$$\forall a, s.Poss(a, s) \supset P(s, do(a, s))$$
$$\forall s_1, s_2, s_3.P(s_1, s_2) \wedge P(s_2, s_3) \supset P(s_1, s_3)$$

It is not hard to show that with this definition, the final two axioms do not have to be postulated, and are in fact logical theorems.

3 The Logic $\mathcal{OL}$

The language of $\mathcal{OL}$ is a modal first-order dialect with equality and function symbols plus a countably infinite set of standard names $\mathcal{N} = \{\#0, \#1, \#2, \ldots\}$ which will serve as our universe of discourse. As discussed in more detail in [Lev84], standard names allow interesting distinctions between *de dicto* and *de re* beliefs (see below for an example). Terms and atomic formulas are defined as usual. A term is called *primitive* if it consists of a function symbol followed by standard names as argument. Similarly a formula is called primitive if it consists of a predicate symbol followed by standard names. Arbitrary formulas of $\mathcal{OL}$ are constructed in the usual way from the atomic formulas, equality, the connectives $\neg$ and $\vee$, the quantifier $\forall$,[5] and the modal operators $\boldsymbol{K}$ and $\boldsymbol{O}$, where $\boldsymbol{K}\alpha$ should be read as "the agent knows α" and $\boldsymbol{O}\alpha$ as "the agent only knows α." *Sentences* are formulas without free variables. A formula is called *objective* if it does not contain any modal operators. Vector notation will be used freely for sequences, for example, $\forall \vec{x}$ for $\forall x_1 \cdots \forall x_k$.

The semantics of $\mathcal{OL}$ is based on the familiar notion of a world, which assigns meaning to the nonlogical symbols of the language. $\mathcal{OL}$ makes the assumption that all worlds have as their universe of discourse the same set which is isomorphic to the standard names, that is, an individual is identified with a unique name. A world is then completely specified by providing the meaning of every primitive term and formula:

Definition 3.1: A *world* w is a function from primitive expressions into $\{0, 1\} \cup \mathcal{N}$, where $w[p] \in \{0, 1\}$ for primitive formulas p, and $w[t] \in \mathcal{N}$ for primitive terms t.

Given a world w, the denotation of an arbitrary ground term t is defined recursively as

$|n|_w = n$, where n is a standard name.

$|f(t_1, \ldots, t_k)|_w = w[f(n_1, \ldots, n_k)]$, where $n_i = |t_i|_w$.

We often write $|\vec{t}|_w$ instead of $\langle |t_1|_w, \ldots, |t_k|_w \rangle$.

While the truth of objective sentences is determined by a single world w, the meaning of sentences of the form $\boldsymbol{K}\alpha$ and $\boldsymbol{O}\alpha$ is defined relative to a *set* of worlds e. $\boldsymbol{K}\alpha$ holds if α is true in all worlds of e. $\boldsymbol{O}\alpha$ holds if $\boldsymbol{K}\alpha$ holds and, in addition, every world that satisfies α is also a member of e. This way e minimizes what is known besides α. e is also called an *epistemic state* and a pair (e, w) is sometimes referred to as a an $\mathcal{OL}$ model.

The semantic rules which determine the truth of a sentence α at a given world w and epistemic state e (denoted as $e, w \models \alpha$) are defined as follows:

$e, w \models F(\vec{t})$	iff	$w[F(\vec{n})] = 1$ and $\vec{n} = \lvert\vec{t}\rvert_w$, where $F(\vec{t})$ is atomic.
$e, w \models t_1 = t_2$	iff	$\lvert t_1\rvert_w = \lvert t_2\rvert_w$
$e, w \models \neg\alpha$	iff	$e, w \not\models \alpha$
$e, w \models \alpha \vee \beta$	iff	$e, w \models \alpha$ or $e, w \models \beta$
$e, w \models \forall x \alpha$	iff	$e, w \models \alpha^x_n$ for all $n \in \mathcal{N}$
$e, w \models \boldsymbol{K}\alpha$	iff	for all $w' \in e$, $e, w' \models \alpha$
$e, w \models \boldsymbol{O}\alpha$	iff	for all w', $w' \in e$ iff $e, w' \models \alpha$

A formula α is valid ($\models_{\text{OL}} \alpha$) iff $e, w \models \alpha$ for all worlds w and all sets of worlds e.[6] We sometimes write $w \models \alpha$ if α is an objective sentence.

Here we only briefly discuss the operators $\boldsymbol{K}$ and $\boldsymbol{O}$. For a detailed discussion of $\mathcal{OL}$, we refer the reader to [Lev90] and [LL9x]. $\boldsymbol{K}$ has the usual properties of the logic *K45* or *weak S5* [HC68] whose characteristic axioms are:

K: $\boldsymbol{K}(\alpha \supset \beta) \supset (\boldsymbol{K}\alpha \supset \boldsymbol{K}\beta)$
4: $\boldsymbol{K}\alpha \supset \boldsymbol{K}\boldsymbol{K}\alpha$
5: $\neg\boldsymbol{K}\alpha \supset \boldsymbol{K}\neg\boldsymbol{K}\alpha$

The Barcan formula ($\forall x \boldsymbol{K}\alpha \supset \boldsymbol{K}\forall x\alpha$) is also valid since we are assuming a fixed universe of discourse [HC68]. While $\boldsymbol{K}$ is very well understood, this is less the case for $\boldsymbol{O}$ except perhaps when $\boldsymbol{O}$ is applied to an objective sentence. Consider an atomic sentence p. It is easy to see that the only epistemic state e where $\boldsymbol{O}p$ is satisfied is $e = \{w \mid w \models p\}$, that is, the set of all worlds where p is true. It is the "iff" in the semantic rule of $\boldsymbol{O}$ which has the effect of maximizing e. As a result, the objective sentences known at e are exactly the logical consequences of p, which captures the idea that p is all that is known. Note that the meaning of $\boldsymbol{O}$ crucially depends on e as well as on the complement of e. In particular, for $\boldsymbol{O}\alpha$ to be true, α has to be known and all worlds not in e have to falsify α. The story becomes much more complicated with arbitrary formulas in the scope of

[5]Other logical connectives like $\wedge$, $\supset$, and $\equiv$ and the quantifier $\exists$ are used freely and are defined in the usual way.

[6]Levesque used so-called *maximal sets* to define validity, a complication we ignore here for simplicity.

O. For example, as shown in [Lev90], using ***O*** it is possible to fully reconstruct and extend Moore's Autoepistemic Logic [Moo85b]. However, in this paper we will not be concerned with such issues, and all the examples used here consider only knowing applied to objective sentences.

[Lev90] presents an axiomatization of $\mathcal{OL}$, which is complete for the propositional case, but was recently shown to be incomplete for the full first-order language [HL95]. An interesting by-product of our work is that, by appealing to second-order logic, a complete axiom system for $\mathcal{OL}$ obtains.

To see the utility of ***O***, consider an agent who knows the standard name for the current temperature, which could be a particular value on some temperature scale. If this is all she knows then it should follow that she knows what the temperature is and that she does not know what the barometric pressure is. In $\mathcal{OL}$ this can be expressed by the following valid sentence, where n is a standard name and both *temperature* and *pressure* are ordinary constants:

$$\boldsymbol{O}(\mathit{temperature} = n) \supset \\ \exists x \boldsymbol{K}(\mathit{temperature} = x) \land \neg\exists y \boldsymbol{K}(\mathit{pressure} = y).$$

Note that the implication would not go through if we replaced ***O*** by ***K***: knowing the temperature does not rule out knowing the barometric pressure as well. We feel that being able to derive facts about what is *not* known without having to state them as premises is a useful feature of the language that becomes even more important when reasoning about knowledge and action.

4 The Logic $\mathcal{AOL}$

There are two ways of understanding what is required to amalgamate the situation calculus and the logic $\mathcal{OL}$: we can view it in terms of extensions to $\mathcal{OL}$, and how the semantics there needs to be modified, or we can view it in terms of extensions to the situation calculus, and how the foundational axioms need to change. Our approach can be seen as taking both views. We begin by proposing a semantics which extends that of $\mathcal{OL}$ by adding actions and applying it to a slightly extended language of the situation calculus to account for knowledge and standard names. We then provide a small set of axioms which are sound and complete for the semantics and which, taken by themselves, can be thought of as foundational axioms for an extended situation calculus.

The language of $\mathcal{AOL}$ will be a dialect of the second-order predicate calculus, like the situation calculus introduced in Section 2. Again we have three sorts: ordinary objects, actions and situations. The constant S_0, the function *do*, and special predicate $\mathit{Poss}(a, s)$ are exactly as before. We will however require two new special predicates, $\mathit{SF}(a, s)$ and $K_0(s)$, a new constant 0 of sort ordinary object, and a new function symbol $\mathit{succ}(x)$, which maps ordinary objects to ordinary objects. The set of (relational and functional) fluents as well as the set of action function symbols is assumed to be finite.

For simplicity, we also make the following restrictions: there are no constants or functions of the situation sort other than S_0 and *do*; action functions do not take situations as arguments; all ordinary object functions other than 0 and *succ* are fluents; and all predicates other than those mentioned above are fluents. Finally, we assume that fluents only have situation terms as arguments in the final position.

Recall that $\mathcal{OL}$ assumes a fixed countably infinite domain of ordinary objects, isomorphic to the set of standard names. We want to use standard names as objects in $\mathcal{AOL}$ as well. However, in order to facilitate the axiomatization later on, we will not represent them using infinitely many distinct predicate calculus constants $^{\#}0, ^{\#}1, \ldots$, but instead construct them using 0 and *succ*. The idea is that 0 will play the role of $^{\#}0$, $\mathit{succ}(0)$ the role of $^{\#}1$, *etc.* (similar to the way numerals are represented in number theory). As before, we refer to the set of standard names (now ground terms) as $\mathcal{N}$.

To deal with knowledge in $\mathcal{AOL}$, the biggest change is that we imagine that in addition to S_0 and its successors, there are an uncountable number of other initial and non-initial situations considered as possible epistemic alternatives. To state what is known in S_0, we use K_0. Informally, taking S_0 to be the situation counterpart to the given world w in $\mathcal{OL}$, K_0 is the counterpart to the given epistemic state e in $\mathcal{OL}$. In other words, $K_0(s)$ is intended to hold if s is a situation considered by the agent in S_0 to be possible. How knowledge changes when performing an action a in situation s is governed by $\mathit{SF}(a, s)$ and $\mathit{Poss}(a, s)$ and will be discussed later in Section 5.

4.1 Semantics

Recall that in $\mathcal{OL}$, there are worlds corresponding to all possible interpretations of the predicate and function symbols (over the domain of standard names). Different applications, of course, will use different subsets as part of the given e, but the complement of e is still relevant because of only knowing. We need the same in $\mathcal{AOL}$ with respect to K_0, but more: we need to allow for all possible interpretations of the predicate and function symbols *after all possible sequences of actions*. That is, to ensure that it is possible to know the initial value of a term or formula without also necessarily knowing its value in successor situations, it is necessary that there be initial situations that agree on the values of all terms and formulas but that have successors that disagree on these values.[7] Thus instead of defining a world as a function from primitive expressions to suitable values as we did in $\mathcal{OL}$, we define a world in $\mathcal{AOL}$ as a function from primitive expressions and sequences of actions to these values. We then define a situation as a pair consisting of a world and a sequence of actions.

Worlds and situations

More precisely, the standard names for objects, as already mentioned, are ground terms involving just 0 and *succ*; the standard names for actions are terms of the form $A(t_1, \ldots, t_n)$ where A is an action function and each t_i is a standard name; there are no standard names for situations, since there will be more situations than expressions in the language. The primitive terms $\mathcal{T}$ are object terms of the form $f(t_1, \ldots, t_n)$ where each t_i is a standard name, and $f(x_1, \ldots, x_n, s)$ is a functional fluent. The primitive formulas $\mathcal{P}$ are atoms of the form $F(t_1, \ldots, t_n)$ where each t_i is a standard name, and $F(x_1, \ldots, x_n, s)$ is a relational fluent, or F is one of *Poss* or *SF*. Note that except for *Poss* and *SF*, primitive expressions are all fluents with the situation argument suppressed.

Let Act^* be the set of all sequences of standard names for actions including the empty sequence ϵ.

Definition 4.1: An $\mathcal{AOL}$ *world* w is a function:

$$w : (\mathcal{P} \times Act^*) \cup (\mathcal{T} \times Act^*) \longrightarrow \{0,1\} \cup \mathcal{N}$$

such that

$$w[p, \vec{a}] \in \{0,1\} \text{ for all } p \in \mathcal{P}.$$
$$w[t, \vec{a}] \in \mathcal{N} \text{ for all } t \in \mathcal{T}.$$

Let $\mathcal{W}$ denote the set of all $\mathcal{AOL}$ worlds.

Definition 4.2: An $\mathcal{AOL}$ *situation* is a pair $(w, \vec{a})$, where $w \in \mathcal{W}$ and $\vec{a} \in Act^*$. An *initial* situation is one where $\vec{a} = \epsilon$.

Definition 4.3: An action model M is a pair $\langle e, w\rangle$, where $w \in \mathcal{W}$ and $e \subseteq \mathcal{W}$.

As in $\mathcal{OL}$, w is taken to specify the actual world, and e specifies the epistemic state as those worlds an agent has not yet ruled out as being the actual one. As we will see below, a situation term s will be interpreted semantically as an $\mathcal{AOL}$ situation $(w, \vec{a})$, consisting of a world and a sequence of actions that have happened so far. A fluent $p(s)$ will be considered true if $w[p, \vec{a}] = 1$.

Because situations cannot have standard names, to interpret formulas with variables, we need to use variable maps. A variable map ν maps object, action, and situation variables into standard names for objects and actions, and into $\mathcal{AOL}$ situations, respectively. In addition, ν assigns relations of the appropriate type[8] to relational variables. For a given ν, ν_o^x denotes the variable map which is like ν except that x is mapped into o.

The meaning of terms

We write $|\cdot|_{M,\nu}$ for the denotation of terms with respect to an action model $M = \langle e, w\rangle$ and a variable map ν. Then

- $|0|_{M,\nu} = 0$;
- $|succ(t)|_{M,\nu} = succ(|t|_{M,\nu})$;
- $|f(\vec{t}, t_s)|_{M,\nu} = w'[f(|\vec{t}|_{M,\nu}), \vec{a}]$, where $f(\vec{t}, t_s)$ is a functional fluent, and $|t_s|_{M,\nu} = (w', \vec{a})$;
- $|A(\vec{t})|_{M,\nu} = A(|\vec{t}|_{M,\nu})$, where $A(\vec{t})$ is an action term;
- $|S_0|_{M,\nu} = (w, \epsilon)$;
- $|do(t_a, t_s)|_{M,\nu} = (w', \vec{a}\cdot a)$, where $|t_s|_{M,\nu} = (w', \vec{a})$, and $|t_a|_{M,\nu} = a$;
- $|x|_{M,\nu} = \nu(x)$, where x is any variable, including predicate variables.

Observe that in a model $M = \langle e, w\rangle$, the only way to refer to a situation that does not use the given world w is to use a situation variable.

The meaning of formulas

We write $M, \nu \models \alpha$ to mean formula α comes out true in action model M and variable map ν:

- $M, \nu \models F(\vec{t}, t_s)$ iff $w'[F(|\vec{t}|_{M,\nu}), \vec{a}] = 1$, where $F(\vec{t}, t_s)$ is a relational fluent, and $|t_s|_{M,\nu} = (w', \vec{a})$;
- $M, \nu \models X(\vec{t})$ iff $|\vec{t}|_{M,\nu} \in \nu(X)$, where X is a relational variable;
- $M, \nu \models Poss(t_a, t_s)$ iff $w'[Poss(|t_a|_{M,\nu}), \vec{a}] = 1$, where $|t_s|_{M,\nu} = (w', \vec{a})$;
- $M, \nu \models SF(t_a, t_s)$ iff $w'[SF(|t_a|_{M,\nu}), \vec{a}] = 1$, where $|t_s|_{M,\nu} = (w', \vec{a})$;
- $M, \nu \models K_0(t_s)$ iff $|t_s|_{M,\nu} = (w', \epsilon)$ and $w' \in e$;
- $M, \nu \models t_1 = t_2$ iff $|t_1|_{M,\nu} = |t_2|_{M,\nu}$;
- $M, \nu \models \neg\alpha$ iff $M, \nu \not\models \alpha$;
- $M, \nu \models \alpha \vee \beta$ iff $M, \nu \models \alpha$ or $M, \nu \models \beta$;
- $M, \nu \models \forall x.\alpha$ iff $M, \nu_o^x \models \alpha$ for all o of the appropriate sort (object, action, situation, relation).

[7] In some applications this generality will not be required.

[8] The type determines the arity and the sort of each argument of the relations the variable ranges over. Since, in our examples, the type will always be obvious from the context, we leave this information implicit.

For sentences α we sometimes write $M \models \alpha$ instead of $M, \nu \models \alpha$.

Validity is defined in the usual way as truth in all models, that is, a formula α is valid in $\mathcal{AOL}$ ($\models_{\text{AOL}} \alpha$) iff for all action models $M = \langle e, w \rangle$ and variable maps ν, $M, \nu \models \alpha$.

4.2 An axiomatization

The first three axioms tell us that the set of objects is isomorphic to the set $\mathcal{N}$ of standard object names. Indeed the formulation resembles the usual second-order definition of the natural numbers, that is, the following three axioms do no more than give us domain closure and unique names axioms for objects.

F1: $\forall x.\, succ(x) \neq 0$
F2: $\forall x, y.\, succ(x) = succ(y) \supset x = y$
F3: $\forall P.\, [P(0) \wedge \forall x(P(x) \supset P(succ(x)))] \supset \forall x. P(x)$

Next we need to say that the actions consist precisely of the primitive actions and that they are all distinct. This can be done in the usual way.

F4: Domain closure and unique names axioms for actions.

Neither *SF* nor *Poss* need special axioms since their meaning is left completely user-defined. The only foundational axiom concerning K_0 is one saying that it only applies to initial situations. To be precise, let

$$Init(s') \doteq \neg\exists a, s. s' = do(a, s).$$

Then we have

F5: $Init(S_0) \wedge \forall s. K_0(s) \supset Init(s)$

Finally we have the job of characterizing the set of situations. As in the original dialect of the situation calculus, we first want to say that any non-initial situation is the result of applying *do* to an initial situation. We use variants of the previous axioms:

F6: $\forall a_1, a_2, s_1, s_2.\, do(a_1, s_1) = do(a_2, s_2) \supset (a_1 = a_2 \wedge s_1 = s_2)$.
F7: $\forall P.\, (\forall s, s'. Init(s) \wedge s \preceq s' \supset P(s')) \supset \forall s. P(s)$

where we have

$$s \preceq s' \doteq \forall R[\ldots \supset R(s, s')]$$

with the ellipsis standing for the conjunction of

$$\forall s_1.\, R(s_1, s_1)$$
$$\forall a, s_1.\, R(s_1, do(a, s_1))$$
$$\forall s_1, s_2, s_3.\, R(s_1, s_2) \wedge R(s_2, s_3) \supset R(s_1, s_3)$$

Then the only remaining job is to characterize the set of initial situations. Looking back at the semantics of $\mathcal{AOL}$, recall that for a correct interpretation of only knowing, we had to insist that there be an initial situation corresponding to any conceivable outcome of the fluents initially and after any sequence of actions. Given the power of second-order logic, it is possible to precisely capture this property axiomatically.[9]

To handle sequences of actions, we begin by introducing an abbreviation $C(s', s)$ intended to say that s' and s involve the same sequence of actions from perhaps different initial states:

$$C(s', s) \doteq \forall R[\ldots \supset R(s', s)]$$

where the ellipsis stands for the conjunction of

$$\forall s_1, s_2.\, Init(s_1) \wedge Init(s_2) \supset R(s_1, s_2)$$
$$\forall a, s_1, s_2.\, R(s_1, s_2) \supset R(do(a, s_1), do(a, s_2)).$$

With this in place, we can use situations s where $S_0 \preceq s$ as a canonical way of talking about sequences of actions.

Suppose that our language contains relational fluents $F_1, \ldots, F_n$.[10] Then we can write our final axiom as:

F8: $\forall s. Init(s) \equiv \forall Q[\ldots \supset Q(s)]$

where the ellipsis stands for

$$\begin{aligned}
&\forall P_1, \ldots, P_{n+2}, \exists s'.\, Q(s') \wedge Init(s') \wedge \\
&\quad \forall \vec{x}_1, \ldots, \vec{x}_n, t, u, a. \\
&\qquad s' \preceq t \wedge S_0 \preceq u \wedge C(t, u) \supset \\
&\qquad\quad F_1(\vec{x}_1, t) \equiv P_1(\vec{x}_1, u) \wedge \ldots \wedge \\
&\qquad\quad F_n(\vec{x}_n, t) \equiv P_n(\vec{x}_n, u) \wedge \\
&\qquad\quad Poss(a, t) \equiv P_{n+1}(a, u) \wedge \\
&\qquad\quad SF(a, t) \equiv P_{n+2}(a, u)
\end{aligned}$$

To see how this axiom works, imagine that $n = 1$, $F_1(x, s)$ is a unary fluent, and ignore *Poss* and *SF*. Then, the set of initial situations is the least set such that for every mapping from sequences of actions (here represented by the u) to sets of objects, there is an initial situation s' such that F_1 holds on exactly that set of objects in the situation t that results from doing those actions starting in s'.

[9]We are indebted to Fangzhen Lin who pointed this out to us.

[10]Recall that apart from the special predicates K_0, *Poss* and *SF*, our language has only finitely many relational and functional fluents. To make things simple, we omit functional fluents completely from this axiom. They require functional variables in the language.

These are all the axioms we need, and we will refer to them collectively as AX from now on, and we let AX $\models \alpha$ stand for "α is logically implied by AX" in ordinary (second-order) logic.

It is not hard to show that the axioms are sound with respect to the semantics of $\mathcal{AOL}$. Moreover, action models are, in a sense, the only models of the axioms. More precisely, one can show that for any arbitrary Tarskian model I of the axioms there is an action model M such that I and M agree on all sentences. The key property is that the objects, actions, and situations of an arbitrary model of AX are isomorphic to the objects, actions, and situations, respectively, of action models. With that we obtain the main result.

Theorem 4.4: *For any* α, $\models_{AOL} \alpha$ *iff* AX $\models \alpha$.[11]

Given this result one may wonder what the point of introducing a non-standard semantics for $\mathcal{AOL}$ is in the first place, that is, why not just use the axioms? Perhaps the strongest argument in favor of the semantics is that it lends independent support to the claim that the axioms are indeed reasonable. The fact that the semantics generalizes that of $\mathcal{OL}$ in a natural way adds further credence to that claim.

5 Knowledge and Action

Before going into details about the connection between knowledge and action in $\mathcal{AOL}$, a few words are in order about how we should envisage using $\mathcal{AOL}$ to model a particular domain of interest.

Instead of simply writing a basic action theory as presented in Section 2 describing what holds in S_0 and after performing actions, the user must now worry about the other initial situations.

Consider, for example, a precondition axiom for an action A. In the ordinary situation calculus, the user would write an axiom of the form

$$\forall s.\ Poss(A, s) \equiv \phi_a(s).$$

where ϕ_a is some formula does not mention *Poss*. The intent of the quantification (given the previous foundational axioms) was for this to hold in S_0 and all its successors. But in $\mathcal{AOL}$, the quantification over *all* situations is much too strong. By virtue of **F8** there will be initial situations where *Poss* and ϕ_a have different truth values, and so the axiom as it stands is false! To achieve the desired effect, the user should write instead

$$\forall s'.\ S_0 \preceq s' \supset [Poss(A, s') \equiv \phi_a(s')].$$

to ensure that the precondition applies to S_0 and its successors. It is then a separate step to assert that the precondition is known, if desired. To do so, the user would write

$$\forall s, s'.\ K_0(s) \wedge s \preceq s' \supset [Poss(A, s') \equiv \phi_a(s')].$$

This ensures that the axiom is considered to hold in any situation initially considered possible and all of its successors. Similar considerations apply to writing successor state axioms. In general, we write initial state axioms, precondition axioms, and successor state axioms all parameterized by the initial situation we wish to consider, and only quantify over successors of that initial situation. We will see an example shortly.

Given a specification of what is known in S_0, the predicates *SF* and *Poss* are then used to characterize what is known in successor situations. Note that the logic itself imposes no constraints on either *SF* or *Poss*; it is up to the user in an application to write appropriate axioms. For *Poss*, these are the precondition axioms; for *SF*, the user must write *sensed fluent axioms*, one for each action type, as discussed in [Lev96]. The idea is that $SF(a, s)$ gives the condition sensed by action a in situation s. So we might have, for example,

$$SF(\mathsf{sonar}, s) \equiv (wdist(s) < 10)$$

as a way of saying that the sonar sensing action in situation s tells the robot whether or not the distance to the wall in s is less than 10 units. In case the action a has no sensing component (as in simple physical actions, like moving), the axiom should state that $SF(a, s)$ is identically TRUE. Having defined *SF* as a predicate, we essentially confine ourselves to sensing truth values. If we want the result of a sense action to be the value of a term such as a sonar measuring the actual distance to the wall, we can do so by simply redefining *SF* as a function and treating TRUE and FALSE as special values returned by *SF*. To keep the presentation simple, however, we ignore this issue here.

With these terms, we can now define $K(s', s)$ as an abbreviation for a formula that characterizes when a situation s' is accessible from an arbitrary situation s:[12]

$$K(s', s) \doteq \forall R[\ldots \supset R(s', s)]$$

where the ellipsis stands for the conjunction of

$$\begin{array}{l} \forall s_1, s_2.\ Init(s_1) \wedge Init(s_2) \wedge K_0(s_2) \supset R(s_2, s_1) \\ \forall a, s_1, s_2.\ R(s_2, s_1) \wedge (SF(a, s_2) \equiv SF(a, s_1)) \wedge \\ \qquad (Poss(a, s_2) \equiv Poss(a, s_1)) \supset \\ \qquad\qquad R(do(a, s_2), do(a, s_1)). \end{array}$$

[11]For reasons of space proofs are generally omitted and deferred to a longer version of this paper [LL98].

[12]We could have defined K, as well as $\preceq$ and C, as a predicate in the language as is usually done, but we have chosen not to simply because we wanted to keep the formal apparatus as small as possible.

Space precludes a detailed analysis of this definition, except to claim that it satisfies the successor state axiom for a predicate K proposed in [SL93] as a solution to the frame problem for knowledge and later reformulated in [Lev96], whose notation we follow here:

Theorem 5.1: *The following is a theorem of $\mathcal{AOL}$:*

$$\begin{aligned}&\forall a, s, s'.\ Poss(a,s) \supset K(s', do(a,s)) \equiv\\&\qquad \exists s''.\ s' = do(a,s'') \wedge K(s'',s) \wedge Poss(a,s'')\\&\qquad\quad \wedge\ [SF(a,s) \equiv SF(a,s'')].\end{aligned}$$

Given K, knowledge can then be defined in a way similar to possible-world semantics [Kri63, Hin62, Moo85a] as truth in all accessible situations. Similarly, only knowing a sentence α at a situation s means that all and only those situations with the same action history as s are accessible. We denote the two forms of knowledge using the following macros, where α may contain the special situation symbol *now*. Let α_s^{now} refer to α with all occurrences of *now* replaced by s. Then

$$\begin{aligned}&\mathtt{Knows}(\alpha, s) \doteq \forall s' K(s', s) \supset \alpha_{s'}^{now}\\&\mathtt{OKnows}(\alpha, s) \doteq \forall s' C(s', s) \supset (K(s', s) \equiv \alpha_{s'}^{now}).\end{aligned}$$

For example, $\mathtt{Knows}(Broken(x, now), S_0)$ stands for $\forall s K(s, S_0) \supset Broken(x, s)$ and should be read as "the agent knows in S_0 that x is (now) broken."

6 An Example

We now turn to an example showing how knowing and only knowing can be combined with actions in $\mathcal{AOL}$.

Imagine a robot that lives in a 1-dimensional world, and that can move towards or away from a fixed wall. The robot also has a sonar sensor that tells it when it gets too close to the wall, say, less than 10 units away. So we might imagine three actions, adv and rev which move the robot one unit towards and away from the wall, and a sonar sensing action. We have a single fluent, $wdist(s)$, which gives the actual distance from the robot to the wall in situation s.

We begin by defining precondition axioms, sensed fluent axioms and successor state axioms, all parameterized by some initial situation s (as discussed in Section 5). Let $ALL(s)$ stand for the conjunction of these formulas:

$$\begin{aligned}&\forall s'.s \preceq s' \supset Poss(\mathsf{adv}, s') \equiv wdist(s') > 0\\&\forall s'.s \preceq s' \supset Poss(\mathsf{rev}, s') \equiv \mathrm{TRUE}\\&\forall s'.s \preceq s' \supset Poss(\mathsf{sonar}, s') \equiv \mathrm{TRUE}\\&\forall s'.s \preceq s' \supset SF(\mathsf{adv}, s') \equiv \mathrm{TRUE}\\&\forall s'.s \preceq s' \supset SF(\mathsf{rev}, s') \equiv \mathrm{TRUE}\\&\forall s'.s \preceq s' \supset SF(\mathsf{sonar}, s') \equiv Close(s')\end{aligned}$$

$$\begin{aligned}&\forall s'.s \preceq s' \supset\\&\quad \forall a.\ wdist(do(a, s')) = z \equiv\\&\qquad a = \mathsf{adv} \wedge z = wdist(s') - 1\\&\qquad \vee\ a = \mathsf{rev} \wedge z = wdist(s') + 1\\&\qquad \vee\ z = wdist(s') \wedge a \neq \mathsf{adv} \wedge a \neq \mathsf{rev}\end{aligned}$$

The formula $Close(s)$ in the above is an abbreviation:

$$Close(s) \doteq wdist(s) < 10.$$

Now we are ready to consider some specifics having to do with what is true initially. Assume the robot is located initially 6 units away from the wall in S_0. For simplicity, we also assume that all of the axioms above are true in S_0, that they are also known in S_0, and this is all that is known. So let Γ_0 be the conjunction of the axioms in **AX** and:

$$wdist(S_0) = 6\ \wedge\ ALL(S_0)\ \wedge\ \mathtt{OKnows}(ALL(now), S_0)$$

With these in hand,[13] we are ready to establish some properties of the relationship between knowledge and action.

1. After reading its sonar sensor, the robot knows that it is close to the wall:

   $$\Gamma_0 \models \mathtt{Knows}(Close, do(\mathsf{sonar}, S_0))$$

   The proof is as follows: Suppose $M \models \Gamma_0$. Further suppose that $M, \nu \models K(s, do(\mathsf{sonar}, S_0))$. By Theorem 5.1, we get that

   $$\begin{aligned}M, \nu \models\ &\exists s'.s = do(\mathsf{sonar}, s')\\&\wedge K_0(s') \wedge [Close(s') \equiv Close(S_0)].\end{aligned}$$

   Since $M \models Close(S_0)$ and

   $$M \models \forall s.K_0(s) \supset (wdist(do(\mathsf{sonar}, s)) = wdist(s))$$

   by virtue of Γ_0, we get that $M, \nu \models Close(s)$. Thus, we have that

   $$M \models \forall s.K(s, do(\mathsf{sonar}, S_0)) \supset Close(s).$$

2. Before reading its sensor, however, the robot does not know if it is close to the wall:

   $$\Gamma_0 \models \neg\mathtt{Knows}(Close, S_0)$$

   The proof is as follows: Suppose $M \models \Gamma_0$. Let w' be the element of $\mathcal{W}$ which is just like w except that $w'[wdist, \epsilon] = 10$. Then, when $\nu(s) = (w', \epsilon)$, we have that $M, \nu \models \neg Close(s)$ and $M, \nu \models ALL(s)$. From the latter and the fact that

   $$M \models \forall s.Init(s) \supset (K_0(s) \equiv ALL(s)),$$

   it follows that $M, \nu \models K_0(s)$. Consequently,

   $$M \models \neg\forall s.K(s, S_0) \supset Close(s).$$

[13] Strictly speaking, we also need axioms for basic arithmetic as needed by the example. For simplicity we ignore this complication here.

3. After reading its sensor and moving closer to the wall, the robot continues to know that the wall is close:

$$\Gamma_0 \models \mathtt{Knows}(Close, do(\text{adv}, do(\text{sonar}, S_0)))$$

The proof is as follows: Suppose $M \models \Gamma_0$. Further suppose that $M, \nu \models K(s, do(\text{adv}, do(\text{sonar}, S_0)))$. By Theorem 5.1, we get that

$$M, \nu \models \exists s'.s = do(\text{adv}, do(\text{sonar}, s')) \wedge K_0(s') \\ \wedge [Close(s') \equiv Close(S_0)].$$

Since $M \models Close(S_0)$ and

$$M \models \forall s.K_0(s) \supset \\ (wdist(do(\text{adv}, do(\text{sonar}, s))) = wdist(s) - 1)$$

by virtue of Γ_0, we get that $M, \nu \models Close(s)$. Thus, we have that

$$M \models \forall s.K(s, do(\text{adv}, do(\text{sonar}, S_0))) \supset \\ Close(s).$$

4. After reading its sensor and moving away from the wall, the robot is still close to the wall, but no longer knows it:

$$\Gamma_0 \models Close(do(\text{rev}, do(\text{sonar}, S_0))) \wedge \\ \neg\mathtt{Knows}(Close, do(\text{rev}, do(\text{sonar}, S_0))).$$

The proof is as follows: Suppose $M \models \Gamma_0$. To show that $M \models Close(do(\text{rev}, do(\text{sonar}, S_0)))$, we need only observe that because of Γ_0, we have that

$$M \models wdist(do(\text{rev}, do(\text{sonar}, S_0))) = 7.$$

To show that

$$M \models \neg\mathtt{Knows}(Close, do(\text{rev}, do(\text{sonar}, S_0))),$$

we begin by letting w' be the element of $\mathcal{W}$ this is just like w except that $w'[wdist, \epsilon] = 9$. Then, when $\nu(s) = (w', \epsilon)$, we have that $M, \nu \models Close(s)$, and so because of Γ_0, we get that $M, \nu \models K_0(s)$. It follows that

$$M, \nu \models K(do(\text{rev}, do(\text{sonar}, s)), \\ do(\text{rev}, do(\text{sonar}, S_0)))$$

and moreover, that

$$M, \nu \models wdist(do(\text{rev}, do(\text{sonar}, s))) = 10.$$

Thus, we have that

$$M, \nu \models \exists s.K(s, do(\text{rev}, do(\text{sonar}, S_0))) \wedge \\ \neg Close(s).$$

5. As for knowledge of the future, we have that the robot knows initially that after it reads its sonar, it will know whether or not it is close to the wall:

$$\Gamma_0 \models \mathtt{Knows}([\mathtt{Knows}(Close, do(\text{sonar}, now)) \vee \\ \mathtt{Knows}(\neg Close, do(\text{sonar}, now))], S_0)$$

The proof is as follows: Suppose $M \models \Gamma_0$. Suppose further that $M, \nu \models K_0(s)$. There are two cases to consider: suppose that $M, \nu \models Close(s)$. Then, by an argument similar to the one for (1) above, we get

$$M, \nu \models \mathtt{Knows}(Close, do(\text{sonar}, s));$$

if on the other hand, $M, \nu \models \neg Close(s)$, by a similar argument we get that

$$M, \nu \models \mathtt{Knows}(\neg Close, do(\text{sonar}, s)).$$

Either way, we have that

$$M \models \forall s.\ K_0(s) \supset [\mathtt{Knows}(Close, do(\text{sonar}, s)) \vee \\ \mathtt{Knows}(\neg Close, do(\text{sonar}, s))]$$

6. As for knowledge of the past, we have for example that after moving closer to the wall, the robot knows that it was at least 1 unit away just before doing that action:

$$\Gamma_0 \models \mathtt{Knows}(\exists s'[now = do(\text{adv}, s') \wedge \\ wdist(s') > 0], do(\text{adv}, S_0))$$

The proof is as follows: Suppose $M \models \Gamma_0$. Suppose further that $M, \nu \models K(s, do(\text{adv}, S_0))$. Then by Theorem 5.1, we have that

$$M, \nu \models \exists s'.s = do(\text{adv}, s') \wedge K_0(s') \wedge Poss(s').$$

From Γ_0, we get that

$$M \models \forall s.K_0(s) \supset [Poss(\text{adv}, s) \equiv (wdist(s) > 0)].$$

Thus, $M, \nu \models \exists s'.s = do(\text{adv}, s') \wedge (wdist(s') > 0)$. So we have that

$$M \models \forall s.K(s, do(\text{adv}, S_0)) \supset \\ [\exists s'.\ s = do(\text{adv}, s') \wedge (wdist(s') > 0)].$$

7 Embedding $\mathcal{OL}$ in $\mathcal{AOL}$

In this section we show that $\mathcal{AOL}$ is a faithful extension of $\mathcal{OL}$ in the following sense. It is possible to translate every sentence α of $\mathcal{OL}$ into a sentence $\alpha[s]$ of $\mathcal{AOL}$, where s is any situation, such that α is valid in $\mathcal{OL}$ iff $\alpha[S_0]$ is a logical consequence of the axioms. The following translation is essentially the same as the one proposed in [Lak96].

Definition 7.1: Given any term or formula ϕ in $\mathcal{OL}$, the corresponding term or formula $\phi[s]$ in $\mathcal{AOL}$, where s is any situation term, is defined as follows.

First, we let * denote the obvious translation from the standard names of $\mathcal{OL}$ into those of $\mathcal{AOL}$ involving 0 and *succ*. For example, $\#3^* = succ(succ(succ(0)))$.

$$\begin{aligned}
&x[s] = x \text{ if } x \text{ is a variable}\\
&n[s] = n^* \text{ if } n \text{ is a standard name}\\
&f(t_1,\ldots,t_n)[s] = f(t_1[s],\ldots,t_n[s],s)\\
&\quad \text{if } f(\vec{t}) \text{ is a term in } \mathcal{OL}\\
&F(t_1,\ldots,t_n)[s] = F(t_1[s],\ldots,t_n[s],s)\\
&\quad \text{if } P(\vec{t}) \text{ is an atomic formula in } \mathcal{OL}\\
&(t_1 = t_2)[s] = (t_1[s] = t_2[s])\\
&(\neg\alpha)[s] = \neg\alpha[s]\\
&(\alpha \vee \beta)[s] = \alpha[s] \vee \beta[s]\\
&(\forall x\alpha)[s] = \forall x\alpha[s]\\
&(\boldsymbol{K}\alpha)[s] = \mathtt{Knows}(\alpha[now], s)\\
&(\boldsymbol{O}\alpha)[s] = \mathtt{OKnows}(\alpha[now], s)
\end{aligned}$$

For example, let $\alpha = \boldsymbol{O}P(a) \supset \neg\exists x\boldsymbol{K}P(x)$. Then

$$\alpha[S_0] = [\forall sC(s,S_0) \supset (K(s,S_0) \equiv P(a(s),s))] \supset \neg\exists x(\forall sK(s,S_0) \supset P(x,s)).$$

Note that we tacitly assume that for each predicate and function symbol in α there is a corresponding fluent of the same name in $\mathcal{AOL}$. Since we are applying the translation only to sentences, one at a time, there is no problem in making this assumption[14]

The embedding of $\mathcal{OL}$ into $\mathcal{AOL}$ is established, roughly, by proving that for every $\mathcal{OL}$ model M there is an "equivalent" action model M', and vice versa. Here "equivalent" means that M satisfies α iff M' satisfies $\alpha[S_0]$ for any α mentioning only fluents in $\mathcal{AOL}$. We denote the set of all fluents of $\mathcal{AOL}$ as $\mathcal{F}$. (Note that *Poss* is not part of $\mathcal{F}$.)

Let us consider informally how to construct, given an $\mathcal{OL}$ model $M = (e, w)$, an equivalent action model $M' = \langle e', w' \rangle$. First, notice that the truth value of $\alpha[S_0]$ is determined by initial situations only. This means that, when mapping M into M', we can simply ignore all actions and non-initial situations. The mapping from M to M' then, roughly, amounts to the following. Let $\bar{e}$ denote the complement of e. Then for each $w^* \in e$ ($\bar{e}$) make sure that e' ($\overline{e'}$) contains all those $\mathcal{AOL}$ worlds w' whose initial situations agree with w^* on $\mathcal{F}$. There is only one complication. It may be the case that there are $\mathcal{OL}$ worlds w_1 and w_2 such that $w_1 \in e$ and $w_2 \in \bar{e}$ and both agree on $\mathcal{F}$. In this case we need to split the $\mathcal{AOL}$ worlds whose initial situations agree with w_1 and w_2 on $\mathcal{F}$ into two nonempty sets and assign them to e' and $\overline{e'}$, respectively. Such a split is always possible and it can easily be arranged based on the truth value of a predicate not occurring in $\mathcal{F}$ (such as *Poss*).

Similarly, one can also show the converse, namely that for every action model M there is an equivalent $\mathcal{OL}$ model M' such that M' satisfies α iff M satisfies $\alpha[S_0]$.

This construction, together with the fact that validity in $\mathcal{AOL}$ is completely characterized by the axioms AX (Theorem 4.4), then leads to the desired embedding of $\mathcal{OL}$.

Theorem 7.2: *Let α be a sentence in $\mathcal{OL}$. Then α is valid in $\mathcal{OL}$ iff $\alpha[S_0]$ is a logical consequence of* AX.

This result then provides us, for the first time, with an axiomatic characterization of the valid sentences of $\mathcal{OL}$.

The careful reader will have noticed that Axiom **F8** needs to vary depending on α, since **F8** must mention explicitly at least all those fluents occurring in α. In the full paper we will show that it is possible to have a fixed axiom system for all sentences of $\mathcal{OL}$ by encoding the infinitely many predicate and function symbols of $\mathcal{OL}$ using only finitely many fluents in $\mathcal{AOL}$.

Apart from this technical issue, it should be noted that the price of the axiomatization of $\mathcal{OL}$ is high in that we need to appeal to second-order logic. Whether there is a first-order axiom system for $\mathcal{OL}$ remains an open question. Note, however, that Halpern and Lakemeyer [HL95] have recently shown that, even if one exists, it cannot be recursive.[15] So chances are that we may have to settle for second-order.

8 Comparison with Lakemeyer's $\mathcal{OLS}$:

As already mentioned, Lakemeyer [Lak96] has also proposed an amalgamation of only knowing and the situation calculus. There are obvious differences between his logic $\mathcal{OLS}$ and $\mathcal{AOL}$. For one, $\mathcal{OLS}$ considers real knowledge rather than belief, that is, whatever is believed in $\mathcal{OLS}$ is also true in S_0. For another, while $\mathcal{OLS}$ has a formal semantics, there is no axiomatization. In addition, there are deeper differences as well, which give rise to anomalies when reasoning about only knowing in $\mathcal{OLS}$ which are not present in $\mathcal{AOL}$.

To start with, only knowing in $\mathcal{OLS}$ does have reasonable properties quite similar to $\mathcal{AOL}$ if it is confined to sentences of the form $\mathtt{OKnows}(\alpha(now), t_s)$, where α contains

[14]There would be a problem if we were to apply the translation to infinite sets of sentences, since $\mathcal{AOL}$ is restricted to finitely many fluents, while $\mathcal{OL}$ is not. We will have a bit more to say about how to deal with this mismatch at the end of this section.

[15]There is a trivial nonrecursive "axiom system", which is simply the set of all valid sentences of $\mathcal{OL}$. The interesting question, of course, is whether there is a system with a finite set of axiom schemas.

no situation terms other than now and t_s is a closed situation term. Intuitively, here we are looking only at what is known about one particular situation, which is very similar to $\mathcal{OL}$ except that we can now ask queries about successor situations as well.

Problems arise when we allow arbitrary sentences as arguments of only knowing in $\mathcal{OLS}$, in particular those where we talk about more than one situation as in precondition axioms. The difficulty is that $\mathcal{OLS}$ models have, in a sense, far fewer situations than there are in $\mathcal{AOL}$. In particular, in $\mathcal{OLS}$ there are only as many initial situations as there are different valuations of the fluents (with the situation argument suppressed). This means that any two distinct initial situations in $\mathcal{OLS}$ must differ in the value of at least one primitive expression. In other words, it is impossible to represent within an $\mathcal{OLS}$ model two different courses of events which have completely identical initial situations and only diverge after some actions have been performed. In addition, *Poss* is handled as a function from situations and actions into $\{0, 1\}$, which further restricts the range of possibilities. For example, there are $\mathcal{OLS}$ models where no actions are possible anywhere.

Without going into any further detail of the formalism, let us consider again the example of Section 6. It is possible to construct an $\mathcal{OLS}$ model M such that M satisfies $ALL(S_0) \wedge wdist(S_0) = 6$, M does not satisfy $ALL(s)$ for any other initial situation s (for example, by choosing $Poss(\mathsf{rev}, s)$ to be false), and S_0 is the only situation epistemically accessible from S_0. Then M satisfies $\mathtt{OKnows}(ALL(now), S_0)$ since S_0 is the only initial situation where ALL holds. In addition, M also satisfies $\mathtt{Knows}(Close, S_0)$ since the distance to the wall is 6 at S_0 and there are no other epistemic alternatives. This is clearly unintuitive since the robot has neither been told what its position is nor has it read its sensors. Moreover, the problem arises precisely because there are not enough situations. In particular, there are no initial situations where ALL holds and the distance to the wall is greater than 9.

We believe that $\mathcal{AOL}$ fixes the problems of $\mathcal{OLS}$ in just the right way.

9 Conclusion

In summary, we have introduced the logic $\mathcal{AOL}$ which amalgamates both the situation calculus and the logic of only knowing $\mathcal{OL}$. Besides a semantics we have provided a sound and complete set of axioms. $\mathcal{AOL}$ is compatible with earlier work on knowledge and action and improves on a previous approach to only knowing in the situation calculus. By way of examples we demonstrated that $\mathcal{AOL}$ allows us to make distinctions which are intuitive and, as far as we know, cannot be handled by other formalisms. Finally, as a side-benefit we obtained a complete axiomatization of $\mathcal{OL}$.

$\mathcal{AOL}$ should be understood as a specification of only knowing within a theory of action. We do not expect $\mathcal{AOL}$ to be implemented in its full generality. On the other hand, note that the second-order situation calculus forms the basis of the high-level control language GOLOG [LRL97], which is used in real robots as in [BCF98]. We believe that $\mathcal{AOL}$ has a role to play in future extensions of GOLOG. Other future work includes extensions of $\mathcal{AOL}$ itself such as a generalization to the multi-agent case.

Acknowledgements

We are indebted to Fangzhen Lin for his idea how to axiomatically characterize the set of all situations in second-order logic.

References

[BCF98] Burgard, W., Cremers, A. B., Fox, D., Hähnel, D., Lakemeyer, G., Schulz, D., Steiner, W., Thrun, S., The Interactive Museum Tour-Guide Robot, to appear: *AAAI-98*.

[dVS94] del Val, A. and Shoham, Y., A Unified View of Belief Revision and Update. *Journal of Logic and Computation* Special Issue on Actions and Processes, **4**, 1994, pp. 797–810.

[HL95] Halpern, J. Y. and Lakemeyer, G., Levesque's Axiomatization of Only Knowing is Incomplete. *Artificial Intelligence* **74**(2), 1995, pp. 381–387.

[HM92] Halpern, J. Y. and Moses, Y. O., A Guide to Completeness and Complexity for Modal Logics of Knowledge and Belief. *Artificial Intelligence*, **54**, 1992, pp. 319–379.

[Hin62] Hintikka, J., *Knowledge and Belief: An Introduction to the Logic of the Two Notions.* Cornell University Press, 1962.

[HC68] Hughes, G. E. and Cresswell, M. J., *An Introduction to Modal Logic*, Methuen and Company Ltd., London, England, 1968.

[Kap71] Kaplan, D., Quantifying In, in L. Linsky (ed.), *Reference and Modality*, Oxford University Press, Oxford, 1971.

[Kri63] Kripke, S. A., Semantical considerations on modal logic. *Acta Philosophica Fennica* **16**, 1963, pp. 83–94.

[Lak95] A Logical Account of Relevance. *Proc. of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, Morgan Kaufmann, 1995, pp. 853–859.

[Lak96] Lakemeyer, G., Only Knowing in the Situation Calculus, *Proc. of the Fifth International Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann, San Francisco, 1996, pp. 14–25.

[LL98] Lakemeyer, G. and Levesque, H. J. $\mathcal{AOL}$: a logic of acting, sensing, knowing, and only knowing. In preparation.

[Lev84] Levesque, H. J., Foundations of a Functional Approach to Knowledge Representation, *Artificial Intelligence*, **23**, 1984, pp. 155-212.

[Lev90] Levesque, H. J., All I Know: A Study in Autoepistemic Logic. *Artificial Intelligence*, North Holland, **42**, 1990, pp. 263–309.

[Lev96] Levesque, H. J., What is Planning in the Presence of Sensing. AAAI-96, AAAI Press, 1996.

[LL9x] Levesque, H. J. and Lakemeyer, G., *The Logic of Knowledge Bases*, Monograph, forthcoming.

[LRL97] H. J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. B. Scherl. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming*, **31**, 59-84, 1997.

[LR94] Lin, F. and Reiter, R., State constraints revisited. *J. of Logic and Computation, special issue on actions and processes*, **4**, 1994, pp. 665–678.

[McC63] McCarthy, J., *Situations, Actions and Causal Laws*. Technical Report, Stanford University, 1963. Also in M. Minsky (ed.), *Semantic Information Processing*, MIT Press, Cambridge, MA, 1968, pp. 410–417.

[Moo85a] Moore, R. C., A Formal Theory of Knowledge and Action. In J. R. Hobbs and R. C. Moore (eds.), *Formal Theories of the Commonsense World*, Ablex, Norwood, NJ, 1985, pp. 319–358.

[Moo85b] Moore, R. C., Semantical Considerations on Nonmonotonic Logic. *Artificial Intelligence* **25**, 1985, pp. 75–94.

[Rei91] Reiter, R., The Frame Problem in the Situation Calculus: A simple Solution (sometimes) and a Completeness Result for Goal Regression. In V. Lifshitz (ed.), *Artificial Intelligence and Mathematical Theory of Computation*, Academic Press, 1991, pp. 359–380.

[Rei93] Reiter, R., Proving Properties of States in the Situation Calculus. *Artificial Intelligence*, **64**, 1993, pp. 337-351.

[SL93] Scherl, R. and Levesque, H. J., The Frame Problem and Knowledge Producing Actions. in *Proc. of the National Conference on Artificial Intelligence (AAAI-93)*, AAAI Press, 1993, 689–695.

Formal Results in Spatial Reasoning

A Canonical Model of the Region Connection Calculus

Jochen Renz
Institut für Informatik
Albert-Ludwigs-Universität
D-79110 Freiburg, Germany

Abstract

Although the computational properties of the Region Connection Calculus RCC-8 are well studied, reasoning with RCC-8 entails several representational problems. This includes the problem of representing arbitrary spatial regions in a computational framework, leading to the problem of generating a realization of a consistent set of RCC-8 formulas. A further problem is that RCC-8 performs reasoning about topological space, which does not have a particular dimension. Most applications of spatial reasoning, however, deal with two- or three-dimensional space. Therefore, a consistent set of RCC-8 formulas might not be realizable within the desired dimension. In this paper we address these problems and develop a canonical model of RCC-8 which allows a simple representation of regions with respect to a set of RCC-8 formulas, and, further, enables us to generate realizations in any dimension $d \geq 1$, even when regions are constrained to be (sets of) polytopes. For three- and higher-dimensional space this can also be done for internally connected regions.

1 INTRODUCTION

The Region Connection Calculus (RCC) (Randell *et al.* 1992) is a topological approach to qualitative spatial representation and reasoning (Cohn 1997) where spatial regions are regular subsets of a topological space. Of particular interest for application purposes is RCC-8, a sub-calculus of RCC that uses eight mutually exhaustive and pairwise disjoint base relations. The computational properties of RCC-8 have been studied thoroughly (Nebel 1995; Renz and Nebel 1997) and efficient reasoning mechanisms were identified.

Despite this, there are still several problems with representing spatial regions within RCC-8. As the calculus is based on topology, spatial regions might be arbitrary subsets of a topological space which are not necessarily analytically describable; therefore, it appears to be difficult to represent spatial regions in a computational framework.

Another representational drawback of using RCC-8 is that a topological space does not have a particular dimension, whereas most applications of qualitative spatial reasoning deal only with two- or three-dimensional space. It might be possible that a set of RCC-8 formulas is consistent but not realizable within a particular dimension. Lemon (1996) gave an example of a set of spatial formulas which is realizable in three dimensional space but not in two dimensional space if regions are internally connected. Lemon used this result to argue that spatial logics like RCC are not an adequate formalism for representing space.

A further problem, which also depends on the ability to represent spatial regions, is finding a realization of a consistent and realizable set of spatial formulas in a particular dimension, instead of just knowing whether the set is realizable or not.

In this paper, we will refer to these representational topics. In order to represent arbitrary spatial regions, it is necessary to have a *canonical model* of RCC-8, i.e., a structure that allows to model any consistent sentence of the calculus. Topological space is of course a canonical model, but, as described above, this does not seem to be very useful for representing regions. Therefore, we will present a new canonical model of RCC-8 that permits a simple representation of spatial regions by reducing them to their necessary topological features with respect to their spatial relations. Based on this model, we will prove that for any consistent set of spatial formulas there are realizations in any dimension $d \geq 1$ when regions are allowed to be internally disconnected. This is still true even when regions are constrained to be sets of polytopes. Actually, internal connectedness of regions is not at all forced in the RCC-theory, so RCC can still be seen as an adequate representation formalism of space. We will also argue that forcing internal connectedness of all regions is too restrictive when dealing with spatial regions. Nevertheless, we will prove that in three- and higher dimensional space every consistent set of spatial formulas can always be realized with internally connected regions. Using the new canonical model for representing spatial regions, it becomes possible to determine realizations of consistent sets of spatial formulas. We will give algorithms for generating realizations of both internally connected and disconnected regions.

The remainder of the paper is structured as follows: In Section 2 we introduce RCC-8 and some basic topological notions. Section 3 sketches the modal encoding of RCC-8 and presents the new canonical model of RCC-8. In Section 4 we give a topological interpretation of this model which is used in Section 5 to prove the results about realizations in particular dimensions. Section 6 describes how models of sets of spatial relations can be determined and how realizations can be generated. In Section 7 we will discuss our results.

2 QUALITATIVE SPATIAL REPRESENTATION WITH RCC

RCC is a topological approach to qualitative spatial representation and reasoning where *spatial regions* are regular subsets of a topological space (Randell *et al.* 1992). Relationships between spatial regions are defined in terms of the relation $C(r,s)$ which is true if and only if the closure of region r is connected to the closure of region s, i.e., if they share a common point. Regions themselves do not have to be internally connected, i.e., a region may consist of different disconnected pieces. The domain of *spatial variables* (denoted as X,Y,Z) is the whole topological space.

RCC-8 (Randell *et al.* 1992) uses a set of eight pairwise disjoint and mutually exhaustive relations, called *base relations*, denoted as DC, EC, PO, EQ, TPP, NTPP, TPP^{-1}, and NTPP^{-1}, with the meaning of *DisConnected, Externally Connected, Partial Overlap, EQual, Tangential Proper Part, Non-Tangential Proper Part*, and their converses. Examples for these relations are shown in Figure 1.

Sometimes it is not known which of the eight base relations holds between two regions, but it is possible to exclude some of them. In order to represent this, unions of base relations can be used.

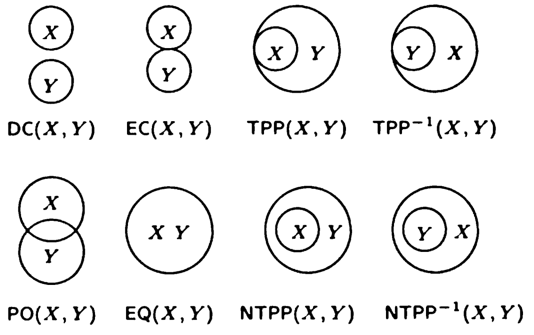


Figure 1: Two-dimensional examples for the eight base relations of RCC-8.

Since base relations are pairwise disjoint, this results in 2^8 different relations. A *spatial formula* $S(X,Y)$ is a relation between two spatial variables, a *spatial configuration* is a set Θ of spatial formulas. Θ is *consistent* if it is possible to find a *realization* of Θ, i.e., a model where every spatial variable is instantiated by a spatial region such that all relations hold between the regions. A consistent instantiation of the spatial variables X,Y,Z will be denoted as $\mathtt{X},\mathtt{Y},\mathtt{Z}$, respectively. Computational properties of reasoning with RCC-8 were studied in (Nebel 1995; Renz and Nebel 1997).

As we will go further into topology, we will define some common topological terms:

Definition 2.1 *Let $\mathcal{U}$ be a set, the* universe. *A* topology *on $\mathcal{U}$ is a family T of subsets of $\mathcal{U}$, with*

1. *if $O_1, O_2 \in T$, then $O_1 \cap O_2 \in T$,*
2. *if $O_i \in T$ for $i \in I$, then $\bigcup O_i \in T$,*
3. *$\emptyset, \mathcal{U} \in T$.*

A topological space *is a pair $\langle \mathcal{U}, T\rangle$. Every subset $O \subset \mathcal{U}$ with $O \in T$ is* open.

If the particular topology T on a set $\mathcal{U}$ is not important, we say that $\mathcal{U}$ is a topological space.

Definition 2.2 *Let $\mathcal{U}$ be a topological space, $M \subset \mathcal{U}$ be a subset of $\mathcal{U}$ and $p \in \mathcal{U}$ be a point in $\mathcal{U}$.*

- *M is* closed *if $\mathcal{U} \setminus M$ is open.*
- *$N \subset \mathcal{U}$ is said to be a* neighborhood *of p if there is an open subset $O \subset \mathcal{U}$ such that $p \in O \subset N$.*
- *p is said to be an* interior point *of M if there is a neighborhood N of p contained in M. The set of all interior points of M is called the* interior *of M, denoted $i(M)$.*

- *p is said to be an* exterior point *of M if there is a neighborhood N of p that contains no point of M. The set of all exterior points of M is called the* exterior *of M, denoted $e(M)$.*
- *p is said to be a* boundary point *of M if every neighborhood N of p contains at least one point in M and one point not in M. The set of all boundary points of M is called the* boundary *of M, denoted $b(M)$.*
- *The* closure *of M is the smallest closed set which contains M, i.e., $M \cup b(M)$.*
- *For any arbitrarily given point $p \in \mathcal{U}$, the family of all neighborhoods of p in $\mathcal{U}$ is called the* neighborhood system *of p in $\mathcal{U}$.*

A neighborhood system $\mathcal{N}$ has the property that every finite intersection of members of $\mathcal{N}$ belongs to $\mathcal{N}$.

It is possible to use any topological space which is a model for the RCC axioms as specified in (Randell *et al.* 1992). Gotts (1996) has shown that every regular connected topological space is a model for the RCC axioms (see also Section 7). So, whenever we refer to a topological space in the remainder of the paper, we mean a regular connected topological space.

3 MODAL ENCODING & CANONICAL MODELS

After making a brief introduction to modal logic, we will introduce the modal encoding of RCC-8 and a canonical model for this encoding.

3.1 MODAL LOGIC & KRIPKE SEMANTICS

Propositional modal logic (Fitting 1993; Chellas 1980) extends classical propositional logic by additional unary *modal operators* $\Box_i$. A common semantic interpretation of modal formulas is the *Kripke semantics* which is based on a *Kripke frame* $\mathcal{F} = \langle W, \mathcal{R}\rangle$ consisting of a set of *worlds* W and a set $\mathcal{R}$ of *accessibility relations* between the worlds, where $R \subseteq W \times W$ for every accessibility relation $R \in \mathcal{R}$. A different accessibility relation $R_{\Box_i} \in \mathcal{R}$ is assigned to every modal operator $\Box_i$. If $u, v \in W$, $R \in \mathcal{R}$, and uRv holds, we say that v is *R-accessible* from u or v is an *R-successor* of u.

A *Kripke model* $\mathcal{M} = \langle W, \mathcal{R}, \pi\rangle$ uses an additional valuation π that assigns each world and each propositional atom a truth value $\{true, false\}$. Using a Kripke model, a modal formula can be interpreted with respect to the set of worlds, the accessibility relations, and the valuation. For example, a propositional atom a holds in a world w of the Kripke model $\mathcal{M}$ (written as $\mathcal{M}, w \Vdash a$) if and only if $\pi(w, a) = true$. An arbitrary modal formula is interpreted according to its inductive structure. A modal formula $\Box_i\varphi$, e.g., holds in a world w of the Kripke model $\mathcal{M}$, i.e., $\mathcal{M}, w \Vdash \Box_i\varphi$, if and only if φ holds in all $R_{\Box_i}$-successors of w. $\mathcal{M}, w \Vdash \neg\Box_i\varphi$ if and only if there is an $R_{\Box_i}$-successor of w where φ does not hold.

$S(X,Y)$	*Model Constraints*	*Entailment Constraints*
DC	$\neg(X \wedge Y)$	$\neg X, \neg Y$
EC	$\neg(\mathbf{I}X \wedge \mathbf{I}Y)$	$\neg(X \wedge Y), \neg X, \neg Y$
PO	—	$\neg(\mathbf{I}X \wedge \mathbf{I}Y), X \rightarrow Y,$ $Y \rightarrow X, \neg X, \neg Y$
TPP	$X \rightarrow Y$	$X \rightarrow \mathbf{I}Y, Y \rightarrow X, \neg X, \neg Y$
TPP^{-1}	$Y \rightarrow X$	$Y \rightarrow \mathbf{I}X, X \rightarrow Y, \neg X, \neg Y$
NTPP	$X \rightarrow \mathbf{I}Y$	$Y \rightarrow X, \neg X, \neg Y$
$NTPP^{-1}$	$Y \rightarrow \mathbf{I}X$	$X \rightarrow Y, \neg X, \neg Y$
EQ	$X \rightarrow Y, Y \rightarrow X$	$\neg X, \neg Y$

Table 1: Encoding of the base relations in modal logic

Different modal operators can be distinguished according to their different accessibility relations. In this paper we are using a so-called S4-operator and an S5-operator. The accessibility relation of an S4-operator is reflexive and transitive, the accessibility relation of an S5-operator is reflexive, transitive, and euclidean.

3.2 MODAL ENCODING OF RCC8

The encoding of RCC-8 in propositional modal logic was introduced by Bennett (1995) and extended in (Renz and Nebel 1997). In both cases the encoding is restricted to regular closed regions. The encoding is based on a set of ***model*** and ***entailment constraints*** for each base relation, where model constraints must be true and entailment constraints must not be true. Bennett encoded these constraints in modal logic by transforming every spatial variable to a propositional atom and introducing an S4-operator **I** which he interpreted as an interior operator (Bennett 1995). In order to distinguish between spatial variables and the corresponding propositional atoms we will write propositional atoms as X, Y. Table 1 displays the constraints for the eight base relations. In order to combine them to a single modal formula, Bennett introduced an S5-operator[1] $\Box$, where $\Box\varphi$ is written for every model constraint φ and $\neg\Box\psi$ for every entailment constraint ψ (Bennett 1995). All constraints of a single base relation are then combined conjunctively to a single modal

[1]Bennett called this a *strong* S5-operator as all worlds are $R_\Box$-accessible from each other, i.e., $R_\Box = W \times W$.

formula. In order to represent unions of base relations, the modal formulas of the corresponding base relations are combined disjunctively. In this way every spatial formula $S(X,Y)$ can be transformed to a modal formula $m_1(S(X,Y))$. Additional constraints $m_2(X)$ are necessary to guarantee that only regular closed regions are used (Renz and Nebel 1997): every region must be equivalent to the closure of its interior, and the complement of a region must be equivalent to its interior.

$$m_2(X) = \Box(\mathsf{X} \leftrightarrow \neg\mathbf{I}\neg\mathbf{I}\mathsf{X}) \wedge \Box(\neg\mathsf{X} \leftrightarrow \mathbf{I}\neg\mathsf{X}).$$

So, any set of spatial formulas Θ can be written as a single modal formula $m(\Theta)$

$$m(\Theta) = \bigwedge_{S(X,Y)\in\Theta} m_1(S(X,Y)) \quad \wedge \bigwedge_{X\in Reg(\Theta)} m_2(X),$$

where $Reg(\Theta)$ is the set of spatial variables of Θ.

3.3 A CANONICAL MODEL OF RCC8

The modal encoding of RCC-8 can be interpreted by Kripke models. As the modal encoding of RCC-8 is equivalent to a set of RCC-8 formulas (Bennett 1995), a canonical model of RCC-8 is a structure that allows a Kripke model for the modal encoding of any consistent set of spatial formulas Θ. In order to obtain a canonical model, we distinguish different levels of worlds of W (Renz and Nebel 1997). A world w is of *level 0* if there is no world $v \neq w$ with $vR_\mathbf{I}w$. A world w is of *level l* if there is a world v of level $l-1$ with $vR_\mathbf{I}w$ and there is no world $u \neq w$ of a level higher than $l-1$ with $uR_\mathbf{I}w$. Based on this hierarchy of worlds, we will define the canonical model of RCC-8.

Definition 3.1 *An* RCC-8-structure $\mathcal{S}_{RCC8} = \langle W_S, \{R_\Box, R_\mathbf{I}\}, \pi_S\rangle$ *has the following properties:*

1. *W_S contains only worlds of level* 0 *and* 1.
2. *For every world u of level* 0 *there are exactly* $2n$ *worlds v of level* 1 *with* $uR_\mathbf{I}v$. *These* $2n+1$ *worlds form an* RCC-8-cluster *(see Figure 2).*
3. *For every world v of level* 1 *there is exactly one world u of level* 0 *with* $uR_\mathbf{I}v$.
4. *For all worlds* $w, v \in W_S$: $wR_\mathbf{I}w$ *and* $wR_\Box v$.

$\mathcal{S}_{RCC8}$ *contains* RCC-8-*clusters with all possible valuations*[2] *with respect to* $R_\mathbf{I}$. *A set of* RCC-8-*clusters* $\mathcal{M} = \langle W, \{R_\Box, R_\mathbf{I}\}, \pi\rangle \subset \mathcal{S}_{RCC8}$ *is an* RCC-8-model *of* $m(\Theta)$ *if* $\mathcal{M}, w \Vdash m(\Theta)$ *for a world* $w \in W$. *In a* polynomial RCC-8-model *the number of worlds is polynomially bounded by the number of regions n.*

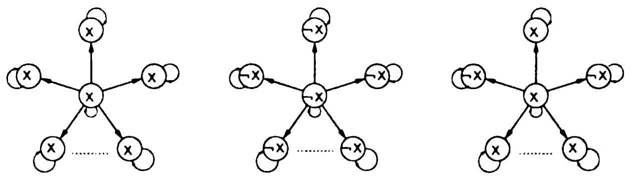


Figure 2: Three possible RCC-8-clusters of $\mathcal{S}_{RCC8}$. Worlds are drawn as circles, the arrows indicate $R_\mathbf{I}$.

In (Renz and Nebel 1997) it was proven that if $m(\Theta)$ is satisfiable, there is a polynomial RCC-8-model $\mathcal{M}$ with $\mathcal{M}, w \Vdash m(\Theta)$ that uses $O(n^2)$ different worlds of level 0 – one world of level 0 for every entailment constraint. So the RCC-8-structure $\mathcal{S}_{RCC8}$ is a canonical model[3] of the modal encoding of RCC-8. In order to obtain a "topological" canonical model for the topological calculus RCC-8, we give in the next section a topological interpretation of RCC-8-models.

4 TOPOLOGICAL INTERPRETATION OF THE CANONICAL MODEL

The modal encoding of RCC-8 was obtained by introducing a modal operator **I** corresponding to the topological interior operator and transferring the topological properties and axioms to modal logic. Using the intended interpretation of **I** as an interior operator, it is unclear how the Kripke models we consider, especially the accessibility relations $R_\Box$ and $R_\mathbf{I}$, can be interpreted topologically. In this section we present a way of topologically interpreting RCC-8-models such that all parts of the models can be interpreted consistently on a topological level. The **I**-operator will not be interpreted as an interior operator, but we will prove that it suffices the intended interpretation.

Because **I** is an S4-operator and because of the additional constraints $m_2(X)$, exactly one of the following formulas is true for every world w of $\mathcal{M}$ and every propositional atom X (see Figure 2).

1. $\mathcal{M}, w \Vdash \mathbf{I}\mathsf{X}$
2. $\mathcal{M}, w \Vdash \mathbf{I}\neg\mathsf{X}$
3. $\mathcal{M}, w \Vdash \mathsf{X} \wedge \neg\mathbf{I}\mathsf{X}$

[2]As the number of spatial variables is countable, the number of RCC-8-clusters is also countable.

[3]The RCC-8-structure does not cover all possible Kripke models of $m(\Theta)$. The goal of a canonical model is just to provide a model for any consistent sentence of a calculus, not to cover all possible models.

Consider a particular world w. Then the set of all spatial variables can be divided into three disjoint sets of spatial variables according to which of the three possible formulas is true in w. Let $\mathcal{I}_w$, $\mathcal{E}_w$, and $\mathcal{B}_w$ be the sets where the first, the second, and the third formula is true in w, respectively, i.e., $\mathcal{M}, w \Vdash \mathsf{IX} \wedge \mathsf{I}\neg\mathsf{Y} \wedge (\mathsf{Z} \wedge \neg\mathsf{IZ})$ for all $X \in \mathcal{I}_w$, $Y \in \mathcal{E}_w$, and $Z \in \mathcal{B}_w$.

When looking at points in a topological space, for every region there are three different kinds of points: interior points, exterior points, and boundary points of a region. If a point is interior or exterior point of a region, there is a neighborhood of the point such that all points of the neighborhood are also inside or outside the region, respectively. If a point is boundary point of a region, every neighborhood contains points inside and points outside the region (see Definition 2.2).

There seems to be a correspondence between worlds and points of a topological space, and between the accessibility relation R_I and topological neighborhoods. In the following lemma we further investigate this correspondence by deriving topological constraints from the modal formulas.

Lemma 4.1 *Let X and Y be two spatial variables of Θ. Depending on which sets $\mathcal{I}_w, \mathcal{E}_w$, or $\mathcal{B}_w$ they are contained in for a world w, the following relations between X and Y are impossible. This has some topological consequences on possible instantiations* $\mathtt{X}, \mathtt{Y}$:

X	Y	*Impossible Relations*	*Consequences*
$\mathcal{I}_w$	$\mathcal{I}_w$	DC, EC	$i(\mathtt{X}) \cap i(\mathtt{Y}) \neq \emptyset$
$\mathcal{I}_w$	$\mathcal{E}_w$	TPP, NTPP, EQ	$i(\mathtt{X}) \cap e(\mathtt{Y}) \neq \emptyset$
$\mathcal{I}_w$	$\mathcal{B}_w$	DC, EC, TPP, NTPP, EQ	$i(\mathtt{X}) \cap b(\mathtt{Y}) \neq \emptyset$
$\mathcal{E}_w$	$\mathcal{E}_w$	–	–
$\mathcal{E}_w$	$\mathcal{B}_w$	TPP^{-1}, NTPP^{-1}, EQ	$e(\mathtt{X}) \cap b(\mathtt{Y}) \neq \emptyset$
$\mathcal{B}_w$	$\mathcal{B}_w$	DC, NTPP, NTPP^{-1}	$b(\mathtt{X}) \cap b(\mathtt{Y}) \neq \emptyset$[4]

Proof: Most entries in the table follow immediately from the encoding of the relations in modal logic. The only more difficult entry is the relation $\mathsf{EC}(X, Y)$ in the third line of the table. This relation is not possible because of the property $\Box(\mathsf{Y} \rightarrow \neg\mathsf{I}\neg\mathsf{IY})$ which states that for any world w that satisfies Y there is a world v with $wR_\mathrm{I}v$ that satisfies IY. As v also satisfies IX, the model constraint of $\mathsf{EC}(X, Y)$ is violated, so this relation is not possible. The topological consequences result from distinguishing the impossible from the possible relations. □

[4]If $\mathsf{PO}(X, Y)$ holds, X and Y do not necessarily have a common boundary point if one of them is not internally connected. However, assuming $b(\mathtt{X}) \cap b(\mathtt{Y}) \neq \emptyset$ in this case does not contradict any spatial formula since RCC-8 is not expressive enough to distinguish different kinds of partial overlap.

It can be seen that when, e.g., **IX** and **IY** hold in a world w, then $\mathtt{X}$ and $\mathtt{Y}$ must have a common interior. So, there is a common interior point of $\mathtt{X}$ and $\mathtt{Y}$ where w can be mapped to.

Theorem 4.2 *Let Θ be a consistent set of spatial formulas, $m(\Theta)$ be the modal encoding of Θ, and $\mathcal{M} = \langle W, \{R_\Box, R_\mathrm{I}\}, \pi\rangle$ be an* RCC-8*-model of $m(\Theta)$. Then there is a function $p : W \mapsto \mathcal{U}$ that maps each world $w \in W$ to a point $p(w) \in \mathcal{U}$ and a function $N : W \mapsto 2^{\mathcal{U}}$ that assigns each world $w \in W$ a neighborhood $N(w)$ of $p(w)$ such that $p(w)$ is in the interior of $\mathtt{X}$ if $\mathcal{M}, w \Vdash \mathsf{IX}$ holds, $p(w)$ is in the exterior of $\mathtt{X}$ if $\mathcal{M}, w \Vdash \mathsf{I}\neg\mathsf{X}$ holds, $p(w)$ is on the boundary of $\mathtt{X}$ if $\mathcal{M}, w \Vdash \mathsf{X} \wedge \neg\mathsf{IX}$ holds, and $p(u) \in N(w)$ if and only if $wR_\mathrm{I}u$ holds.*[5]

Proof: Let w be a world of W and $\mathcal{I}_w, \mathcal{E}_w$, and $\mathcal{B}_w$ be the corresponding sets of spatial variables. We assume that there is a realization of Θ such that there is at least one point in the topological space that is in the interior of every $\mathtt{X}$, in the exterior of every $\mathtt{Y}$, and on the boundary of every $\mathtt{Z}$ simultaneously ($X \in \mathcal{I}_w$, $Y \in \mathcal{E}_w$, $Z \in \mathcal{B}_w$). It follows from Lemma 4.1 that this is true for every pair of regions. As RCC-8 permits only binary constraints between spatial variables and regions are allowed to be internally disconnected, this assumption holds. We further assume that p maps w to one of these points.

Because of Definition 2.2, there must be neighborhoods $N_X(w)$ and $N_Y(w)$ of $p(w)$ for every $X \in \mathcal{I}_w$ and every $Y \in \mathcal{E}_w$ such that $N_X(w)$ is in the interior of $\mathtt{X}$ and $N_Y(w)$ is disjoint with $\mathtt{Y}$. Also, for every $Z \in \mathcal{B}_w$, every neighborhood $N_Z(w)$ of $p(w)$ contains points inside and outside $\mathtt{Z}$. All these neighborhoods are members of the neighborhood system of $p(w)$, so their intersection $N(w)$ is also a neighborhood of $p(w)$ where all R_I-successors of w can be mapped to. □

Using the above defined functions p and N, $\mathcal{M}, w \Vdash \mathsf{IX}$ can be interpreted as "there is a neighborhood $N(w)$ of $p(w)$ such that all points of $N(w)$ are in $\mathtt{X}$". This obeys the intended interpretation of **I** as an interior operator, as $\mathcal{M}, w \Vdash \mathsf{X}$ means that $p(w)$ is in $\mathtt{X}$ and $\mathcal{M}, w \Vdash \mathsf{IX}$ means that $p(w)$ is in the interior of $\mathtt{X}$.

The function N, as defined in Theorem 4.2, can be replaced by any function $N' : W \mapsto 2^{\mathcal{U}}$, with $N'(w) \subseteq N(w)$ for all $w \in W$, if $N'(w)$ is a member of the neighborhood system of $p(w)$. p has to be changed accordingly. In particular, we will regard in the following

[5]The properties for $R_\Box$ ($p(u) \in \mathcal{U}$ if $wR_\Box u$ holds and $p(w) \in \mathcal{U}$) can be omitted as we already defined N and p such that only points of $\mathcal{U}$ are used.

all neighborhoods as d-dimensional spheres.

In order to make the following argumentation easier, a world mapped to an interior point of $\mathtt{X}$ is denoted *interior world* of X, a world mapped to an exterior point of $\mathtt{X}$ *exterior world* of X, and a world mapped to a boundary point of $\mathtt{X}$ *boundary world* of X. Accordingly, a region is called *interior, exterior* or *boundary region* of a world. In particular, a world w with $\mathcal{M}, w \Vdash \mathsf{IX}$ is an interior world of X, a world w with $\mathcal{M}, w \Vdash \mathsf{I}\neg\mathsf{X}$ is an exterior world of X, and a world w with $\mathcal{M}, w \Vdash \mathsf{X} \wedge \neg\mathsf{IX}$ is a boundary world of X.

5 RCC8-MODELS AND THE DIMENSION OF SPACE

In the previous section we have shown how the RCC-8-models introduced in Section 3 can be mapped to topological space, but we still have no information about the dimension of the topological space. In this section we investigate the influence of dimension on the possibility to map the RCC-8-models to the topological space, i.e., which dimension is required in order to find a realization of a consistent set of spatial formulas Θ. We will start with proving that for any RCC-8-model there is a realization in two-dimensional space. It is sufficient to prove this only for sets of base relations as every realization of Θ uses only base relations.[6] For this proof it is important to keep in mind that regions do not have to be internally connected, i.e., they might consist of different disconnected pieces. It will turn out that our proof leads to realizations in any dimension $d \geq 1$. Finally, for three- and higher-dimensional space we will prove that every consistent set Θ can also be realized with internally connected regions.

For the following examination we restrict regions to be sets of d-dimensional polytopes. Sets are required since regions might consist of several disconnected pieces where each piece is a single polytope. This restriction will be lifted later and the results can be generalized to arbitrary regular regions.

Let $\boldsymbol{\Theta}$ be a consistent set of spatial formulas and $\mathcal{M}$ be an RCC-8-model of $m(\Theta)$, the modal encoding of $\boldsymbol{\Theta}$. Suppose that only two-dimensional regions are permitted, i.e., the topological space is a two-dimensional plane $\mathcal{U}$. All worlds of $\mathcal{M}$ are mapped to points of $\mathcal{U}$ as specified in Theorem 4.2. The general intuition of the proof is that every RCC-8-cluster, i.e., every world of level 0 together with its $R_{\mathbf{I}}$-successors is mapped to an independent neighborhood such that each neighborhood can be placed on an arbitrary but distinct position on the plane. Each neighborhood will then be extended to different closed sets that form the pieces of the spatial regions. In the following we will study the requirements neighborhoods have to meet in order to guarantee two-dimensional realizations.

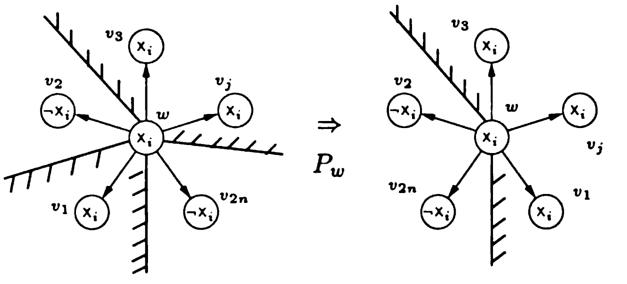


Figure 3: Permutation P_w of the $R_{\mathbf{I}}$-successors of a world w. The solid line indicates the boundary of $\mathtt{X}$, the hashed region the interior of $\mathtt{X}$.

For every spatial variable X_i $(1 \leq i \leq n)$ and every world w of level 0, we define a *region vector* $r_i^w = (r_{i,1}^w, \ldots, r_{i,2n}^w)$ that represents the affiliation of the $2n$ $R_{\mathbf{I}}$-successors of w to X_i, i.e., $r_{i,j}^w = 1$ if $\mathcal{M}, v_j \Vdash \mathsf{X}_i$ and $r_{i,j}^w = 0$ if $\mathcal{M}, v_j \not\Vdash \mathsf{X}_i$ where v_j is the jth $R_{\mathbf{I}}$-successor of w. Since in the two-dimensional case the neighborhood $N(w)$ is a disc, we suppose that the points $p(v_j)$ corresponding to the $R_{\mathbf{I}}$-successors v_j of w are clock-wisely ordered within the disc according to j. If $p(w)$ is a boundary point of $\mathtt{X}_i$, some values of r_i^w are 1 and some are 0. Otherwise all values of r_i^w are either 1 (if $p(w)$ is contained in $\mathtt{X}_i$) or 0 (if $p(w)$ is not contained in $\mathtt{X}_i$).

Lemma 5.1 *If for every world w of level 0 there is a permutation P_w of the values of r_i^w such that $(r_{i,P_w(1)}^w, \ldots, r_{i,P_w(2n)}^w)$ is a bitonic sequence for all $1 \leq i \leq n$, then the neighborhoods $N(w)$ can be placed in a two-dimensional plane such that all spatial relations are satisfied within the neighborhoods.*

Proof: If r_i^w is a bitonic sequence, i.e., the values of r_i^w are in a form $0^e 1^f 0^g$ or $1^e 0^f 1^g$ for $e, f, g \geq 0$, and $p(w)$ is a boundary point of $\mathtt{X}_i$, then the mappings of the worlds of level 1 corresponding to the values of r_i^w can be separated into points inside $\mathtt{X}_i$ and points outside $\mathtt{X}_i$ by at most two line segments meeting at $p(w)$ (see Figure 3). These line segments can be regarded as the part of the boundary of $\mathtt{X}_i$ which is inside $N(w)$. So, neighborhoods can be separated in an interior and an exterior part of a region by a one-dimensional boundary. Therefore all neighborhoods can be placed in a two-dimensional plane. As the per-

[6]The relation EQ can be omitted as any pair of spatial variables X and Y with $\mathsf{EQ}(X, Y)$ can be combined to a single spatial variable.

mutation of the R_I-successors has no influence on the relations between the regions, all spatial relations between the regions hold within the neighborhoods. □

Actually, a permutation as described in the previous lemma is not necessary to guarantee two-dimensional realizations. A region might look as shown on the left of Figure 3, but in this case we restrict the shape and the internal connection of the regions by the neighborhoods we are using which is not at all desirable. However, a permutation as described in Lemma 5.1 is necessary for one-dimensional realizations and realizations with internally connected regions.

Since a permutation P_w is only necessary for boundary worlds, we will in the following regard only those particular RCC-8-models $\mathcal{M}$ for which only those worlds are boundary worlds of regions which are explicitly forced to be boundary worlds of these regions by the constraints. Therefore, we have to take a closer look at which worlds are introduced as boundary worlds of some regions by the entailment constraints, and which worlds are forced to be boundary worlds of regions by the constraints. As a world w of level 0 is forced to be a boundary world of X if $\mathcal{M}, w \Vdash \mathsf{X}$ and $\mathcal{M}, v \not\Vdash \mathsf{X}$ hold for a world v with wR_Iv, we have to find out which of the model and entailment constraints force $\mathcal{M}, w \Vdash \mathsf{X}$ if $\mathcal{M}, v \not\Vdash \mathsf{X}$ holds or force $\mathcal{M}, v \not\Vdash \mathsf{X}$ if $\mathcal{M}, w \Vdash \mathsf{X}$ holds.

Proposition 5.2 *Boundary worlds are introduced only by the following relations (see Table 1):*

1. $\mathsf{EC}(X,Y)$: $\neg\Box(\neg(\mathsf{X} \wedge \mathsf{Y}))$ *introduces a boundary world of* X *and* Y *because of* $\Box(\neg(\mathbf{I}\mathsf{X} \wedge \mathbf{I}\mathsf{Y}))$.
2. $\mathsf{TPP}(X,Y)$: $\neg\Box(\mathsf{X} \rightarrow \mathbf{I}\mathsf{Y})$ *introduces a boundary world of* X *and* Y *because of* $\Box(\mathsf{X} \rightarrow \mathsf{Y})$.
3. $\mathsf{TPP}^{-1}(X,Y)$: $\neg\Box(\mathsf{Y} \rightarrow \mathbf{I}\mathsf{X})$ *introduces a boundary world of* X *and* Y *because of* $\Box(\mathsf{Y} \rightarrow \mathsf{X})$.

Apart from the above worlds that are introduced as boundary worlds of particular regions, worlds can also be forced to be boundary worlds of other regions.

Proposition 5.3 *A world* w *is forced to be a boundary world of* X *only with the following constraints:*

1. $\Box(\mathsf{X} \rightarrow \mathsf{Y})$: *If* w *is a boundary world of* Y *and* X *is true in* w, *then* w *must also be a boundary world of* X.
2. $\Box(\mathsf{Y} \rightarrow \mathsf{X})$: *If* w *is a boundary world of* Y *and* $\neg$X *is true in an* R_I*-successor of* w, *then* w *must also be a boundary world of* X.
3. $\Box(\neg(\mathbf{I}\mathsf{X} \wedge \mathbf{I}\mathsf{Y}))$: *If* X *and* Y *are true in* w, *then* w *must be a boundary world of* X *and* Y.

For the constraints $\Box(\mathsf{X} \rightarrow \mathsf{Y})$ and $\Box(\mathsf{Y} \rightarrow \mathsf{X})$, w must already be a boundary world of some other region, so w must be introduced by one of the relations $\mathsf{EC}(X,Y)$, $\mathsf{TPP}(X,Y)$, or $\mathsf{TPP}^{-1}(X,Y)$. If w is forced to be a boundary world of X and Y with the constraint $\Box(\neg(\mathbf{I}\mathsf{X} \wedge \mathbf{I}\mathsf{Y}))$, then X and Y must both be true in w. This can only be forced when there is a $Z_1 \in Reg(\Theta)$ with $\mathsf{TPP}(Z_1,X)$ and Z_1 is true in w, a $Z_2 \in Reg(\Theta)$ with $\mathsf{TPP}(Z_2,Y)$ and Z_2 is true in w, and w is a boundary world of Z_1 and Z_2 introduced by $\mathsf{EC}(Z_1,Z_2)$. So, in any case when a world is forced to be a boundary world of some region it must already be a boundary world of other regions introduced as described in Proposition 5.2.

We will now have a look at how regions must be related in order to force a world to be a boundary world of these regions using the constraints of Proposition 5.3. Suppose that w is a boundary world of X and Y introduced by either $\mathsf{EC}(X,Y)$ or $\mathsf{TPP}(X,Y)$.[7] We will write $X|Y$ in order to express that we can either use X or Y but always the same. With one of the following constraints it can be forced that w is also a boundary world of $Z \neq X,Y$ (v is an R_I-successor of w):

$$\Box(\mathsf{Z} \rightarrow (\mathsf{X}|\mathsf{Y})) \text{ and } \mathcal{M}, w \Vdash \mathsf{Z} \quad (\rightsquigarrow \mathsf{TPP}(Z, X|Y))$$
$$\Box(\neg(\mathbf{I}\mathsf{Z} \wedge \mathbf{I}(\mathsf{X}|\mathsf{Y}))) \text{ and } \mathcal{M}, w \Vdash \mathsf{Z} \quad (\rightsquigarrow \mathsf{EC}(Z, X|Y))$$
$$\Box((\mathsf{X}|\mathsf{Y}) \rightarrow \mathsf{Z}) \text{ and } \mathcal{M}, v \Vdash \neg\mathsf{Z} \quad (\rightsquigarrow \mathsf{TPP}^{-1}(Z, X|Y))$$

$\mathcal{M}, w \Vdash \mathsf{Z}$ is forced with the following constraint:

$$\Box(\mathsf{U} \rightarrow \mathsf{Z}) \text{ and } \mathcal{M}, w \Vdash \mathsf{U} \quad (\rightsquigarrow \mathsf{TPP}(U, Z))$$

$\mathcal{M}, v \Vdash \neg\mathsf{Z}$ is forced with the following constraints:

$$\Box(\mathsf{Z} \rightarrow \mathsf{U}) \text{ and } \mathcal{M}, v \Vdash \neg\mathsf{U} \quad (\rightsquigarrow \mathsf{TPP}^{-1}(U, Z))$$
$$\Box(\neg(\mathbf{I}\mathsf{Z} \wedge \mathbf{I}\mathsf{U})) \text{ and } \mathcal{M}, v \Vdash \mathsf{U} \quad (\rightsquigarrow \mathsf{EC}(U, Z))$$

When we compose these relations (written as $\circ$), we obtain the possible relations between U and $X|Y$.

$R(U,Z)$	$S(Z,X\|Y)$	$(R \circ S)(U,X\|Y)$
TPP	TPP	TPP, NTPP
TPP	EC	DC, EC
TPP^{-1}	TPP^{-1}	TPP^{-1}, NTPP^{-1}
EC	TPP^{-1}	DC, EC

As w is a boundary world of X and Y, $\mathsf{DC}(U,X|Y)$ and $\mathsf{NTPP}(U,X|Y)$ are not possible together with $\mathcal{M}, w \Vdash \mathsf{U}$, and $\mathsf{NTPP}^{-1}(U,X|Y)$ is not possible together with $\mathcal{M}, v \Vdash \neg\mathsf{U}$. In order to force $\mathcal{M}, w \Vdash \mathsf{U}$,

[7]We omit $\mathsf{TPP}^{-1}(X,Y)$ as $\mathsf{TPP}^{-1}(X,Y) = \mathsf{TPP}(Y,X)$ and the order is not important.

there must be a sequence of spatial variables U_i with $\mathsf{TPP}(U_1, U)$, $\mathsf{TPP}(U_{i+1}, U_i)$, until there is a U_m that is equal to X or Y, so $\mathsf{TPP}(X, Z)$ or $\mathsf{TPP}(Y, Z)$ must hold. In order to force $\mathcal{M}, v \Vdash \neg \mathsf{U}$, there must be a sequence of spatial variables U_i with $\mathsf{TPP}^{-1}(U_1, U)$, $\mathsf{TPP}^{-1}(U_{i+1}, U_i)$, and $\mathsf{EC}(U_j, U_{j-1})$ and $\mathcal{M}, v \Vdash \mathsf{U}_j$ must hold. In order to force $\mathcal{M}, v \Vdash \mathsf{U}_j$, there must be a sequence of TPP-related spatial variables, as described above, until one of them is equal to X or Y, so $\mathsf{EC}(Z, X)$ or $\mathsf{EC}(Z, Y)$ must hold. This results in only three different possibilities of how w is forced to be a boundary world of Z if w was introduced as a boundary world of X and Y.

a. $\mathsf{TPP}(X, Y), \mathsf{TPP}(X, Z)$, and $\mathsf{TPP}(Z, Y)$ hold.

b. $\mathsf{EC}(X, Y), \mathsf{TPP}(X, Z)$, and $\mathsf{EC}(Z, Y)$ hold.

c. $\mathsf{EC}(X, Y), \mathsf{TPP}(Y, Z)$ and $\mathsf{EC}(Z, X)$ hold.

As different spatial variables Z_i, Z_j, for which w is forced to be a boundary world of, all have the boundary world w in common, only the relations $\mathsf{EC}, \mathsf{PO}, \mathsf{TPP}$, or TPP^{-1} can hold between them.

We have shown that only those worlds are boundary worlds which are introduced as boundary worlds of some regions by the entailment constraints, and, further, that other regions are only forced to be boundary regions of these worlds when they are related in a particular way. This will be used in the following lemma.

Lemma 5.4 *Let $\mathcal{M}$ be an* RCC-8-*model. Then two different types of R_{I}-successors are sufficient for every world w of level 0.*

Proof: If w is not a boundary world of some region, all R_{I}-successors of w satisfy exactly the same formulas as w. Otherwise, w is introduced as a boundary world by either $\mathsf{EC}(X, Y)$ or $\mathsf{TPP}(X, Y)$ (see Proposition 5.2). Let w be forced to be a boundary world of the spatial variables Z_i. For $\mathsf{EC}(X, Y)$, some of the R_{I}-successors of w satisfy X but not Y, and some satisfy Y but not X, the others neither satisfy X nor Y. For all Z_i with $\mathsf{TPP}(X, Z_i)$ and $\mathsf{EC}(Z_i, Y)$ and all Z_j with $\mathsf{TPP}(Y, Z_j)$ and $\mathsf{EC}(Z_j, X)$, all R_{I}-successors of w satisfy Z_i if they satisfy X and satisfy Z_j if they satisfy Y. So all R_{I}-successors of w that satisfy X satisfy the same formulas, and all R_{I}-successors of w that satisfy Y satisfy the same formulas. For the R_{I}-successors v of w which do not satisfy X or Y, there are only two requirements: if $\mathsf{TPP}(Z_{k'}, Z_k)$ holds then Z_k must be true in v whenever $\mathsf{Z}_{k'}$ is true in v; if $\mathsf{EC}(Z_k, Z_{k'})$ holds then Z_k and $\mathsf{Z}_{k'}$ must not both be true in v. However, there is no constraint that forces the existence of these worlds, so it can be assumed that all R_{I}-successors

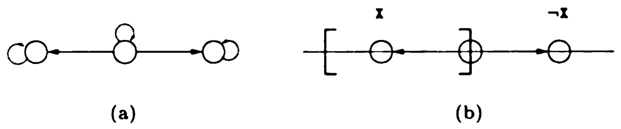


Figure 4: (a) shows a reduced RCC-8-cluster of the reduced RCC-8-structure. (b) shows how a neighborhood can be placed in one-dimensional space. The two brackets indicate a one-dimensional region X where the neighborhood represents a boundary point of X.

of w satisfy either X or Y. As the respective worlds all satisfy the same formulas, two different kinds of R_{I}-successors of the boundary world w introduced by $\mathsf{EC}(X, Y)$ are sufficient.

For $\mathsf{TPP}(X, Y)$, all R_{I}-successors of w that satisfy X also satisfy Y, all R_{I}-successors of w that do not satisfy Y also do not satisfy X, and some R_{I}-successors of w satisfy Y but not X. For all Z_i with $\mathsf{TPP}(X, Z_i)$ and $\mathsf{TPP}(Z_i, Y)$, all R_{I}-successors of w satisfy Z_i if they satisfy X. For the R_{I}-successors of w that satisfy Y but not X, there is only one requirement, namely, that Z_k must be true whenever $\mathsf{Z}_{k'}$ is true in these worlds for any two spatial variables $Z_k, Z_{k'}$ with $\mathsf{TPP}(X, Z_k), \mathsf{TPP}(Z_{k'}, Y)$ and $\mathsf{TPP}(Z_k, Z_{k'})$. However, there is again no constraint that forces the existence of these worlds, so it can be assumed that all R_{I}-successors of w satisfy X if they satisfy Y. □

Whether a boundary world w is introduced by $\mathsf{EC}(X, Y)$ or by $\mathsf{TPP}(X, Y)$, in both cases two different kinds of R_{I}-successors are sufficient. Thus, by grouping together the respective R_{I}-successors for every world w of level 0 of $\mathcal{M}$, we can always find a permutation of the worlds of level 1 such that r^w is a bitonic sequence for all regions.

Instead of having $2n$ R_{I}-successors for every world of level 0 from which we know that they belong to only two different types, it is sufficient two use only two R_{I}-successors for every world of level 0. This leads to a very simple canonical model shown in Figure 4a. We call this a *reduced* RCC-8-*structure* and the corresponding models *reduced* RCC-8-*models*. They are defined in the same way as in Definition 3.1 except that we have exactly two worlds of level 1 instead of $2n$ worlds.

We can now apply Lemma 5.1 and place all neighborhoods independently on the plane while all relations between spatial regions hold within the neighborhoods. Thereby, neighborhoods corresponding to non-boundary worlds are homogeneous in the sense that all points within one of these neighborhoods have the same topological properties. Neighborhoods corresponding to a boundary world w consist of two homogeneous parts corresponding to the two R_I-successors of w. These two parts are divided by the common boundary of the boundary regions of w (see Figure 5a).

In order to obtain a realization, we have to find regions such that the relations between them hold in the whole plane and not just within the neighborhoods. Since regions do not have to be internally connected, it is possible to compose every region out of pieces resulting from the corresponding neighborhoods, i.e., for every neighborhood a region is affiliated with, we generate a piece of that region. As the neighborhoods are open sets and regions as well as their pieces must be regular closed sets, we have to *close* every neighborhood, i.e., find a closed set X^w for every region X and every neighborhood $N(w)$ with $\mathcal{M}, w \Vdash X$ such that all relations hold between the regions composed of the pieces. As all neighborhoods are independent of each other, we only have to make sure that the relations of the different pieces corresponding to a single neighborhood do not violate the relations of the compound regions. This can be done independently for every neighborhood.

Consider a particular neighborhood $N(w)$. If w is not a boundary world, then only the relations PO, TPP, NTPP, and their converse are possible between the regions affiliated with $N(w)$, since they share $N(w)$ as their common interior. For closing the neighborhood $N(w)$, all pieces must fulfill the "part of" relations whereas the PO relations cannot be violated as long as the corresponding pieces have a common interior.

One possibility to fulfill the "part of" relations is using a hierarchy H_Θ of the regions, where a region X is of level $H_\Theta(X) = 1$ if there is no region Y which is part of X. A region X is of level $H_\Theta(X) = k$ if there is a region Y of level $H_\Theta(Y) = k - 1$ which is part of X and if there is no region Z which is part of X and has a higher level than $H_\Theta(Z) = k - 1$.[8] The pieces of all regions affiliated with $N(w)$ must then be chosen according to H_Θ, i.e., pieces of regions of the same level are equal for this particular neighborhood and are non-tangential proper part of all pieces of regions of a higher level. We choose the single pieces to be rectangles.

If w is a boundary world, the boundary regions of w are only affiliated with one part of $N(w)$ and their pieces must share the common boundary. Therefore, both parts of $N(w)$ must be closed separately according to H_Θ (see Figure 5c). In the same way as for two-dimensional space, neighborhoods can be placed in any higher dimensional space and closed therein according to H_Θ. As the three points corresponding to a world w of level 0 and its two R_I-successors can always be aligned, $N(w)$ can also be placed on a line. Thus, all neighborhoods can be placed independently in a one-dimensional space and closed as intervals according to H_Θ (see Figure 4b).

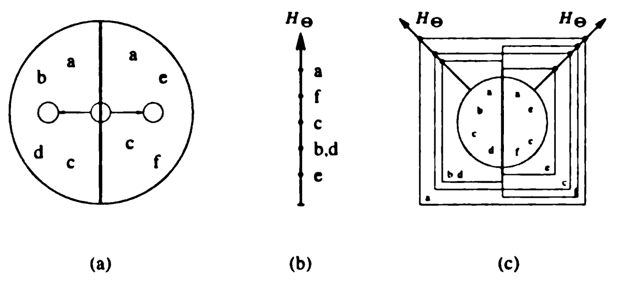


Figure 5: (a) shows the two-dimensional neighborhood of a boundary world which is divided into two parts by the common boundary of the boundary regions b, d, e, and f. (b) shows a possible hierarchy H_Θ of regions. In (c) the neighborhood is closed with respect to H_Θ.

Theorem 5.5 *Every consistent set of spatial formulas can be realized in any dimension $d \geq 1$ where regions are (sets of) d-dimensional polytopes.*

So far all regions consist of as many pieces as there are neighborhoods affiliated with them, i.e., $O(n^2)$ many pieces for every region. We can further show that for three- and higher-dimensional space all regions can also be realized as internally connected. For this we construct a $d+1$-dimensional realization of internally connected regions by connecting all pieces of the same regions of a d-dimensional realization of internally disconnected regions.

Theorem 5.6 *Every consistent set Θ of spatial formulas can be realized with internally connected regions in any dimension $d \geq 3$ where regions are polytopes.*

Proof: Suppose that Θ is consistent. With the following construction we obtain a three-dimensional realization of internally connected regions starting from a two-dimensional realization. (1a) Place all neighborhoods on a circle in the plane determined by the x- and y-axes such that the common boundary of every neighborhood corresponding to a boundary world points to the center of the circle, the z-axis (see Figure 6a). (1b) Close all neighborhoods according to the hierarchy H_Θ such that all pieces of regions are rectangles. (2a) Pro-

[8]This corresponds to the finish time of depth-first search for each vertex of a graph G_Θ where regions are vertices V_Θ and "part of" relations are directed edges E_Θ, computable in time $O(V_\Theta + E_\Theta)$ (Cormen *et al.* 1990, p.477ff)

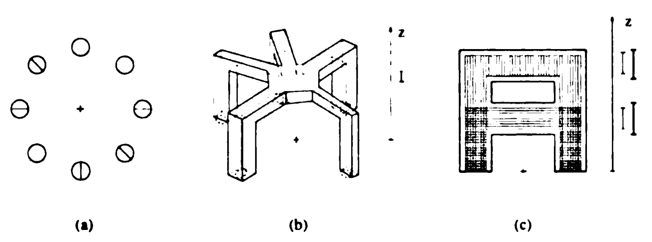


Figure 6: Construction of the three-dimensional realization. (a) placing the two-dimensional neighborhoods on a circle. (b) connecting the pieces of a region on a particular level. (c) connecting the pipes of a region (bold line) that contains the vertically and the horizontally hashed regions.

ceed from this two-dimensional realization according to H_Θ by first choosing pairwise distinct intervals on the positive z-axis for every region with $H_\Theta = 1$, i.e., for the regions that do not contain any other region. (2b) Build a pipe parallel to the z-axis for every piece of these regions starting from the plane (z=0) up to the endpoint of the corresponding interval. (2c) Connect the pipes of the same region within the range of the corresponding interval using pipes pointing to the z-axis (see Figure 6b). (3) Next the regions with $H_\Theta = 2$ are connected, i.e., those regions that only contain already connected regions. To do this, (3a) choose intervals on the z-axis for these regions such that the intervals contain all intervals of the contained regions but do not overlap with any other interval. (3b) Build a pipe for every piece up to the endpoint of the corresponding interval with the largest z-value, and (3c) connect the pipes of every region within the range of all corresponding intervals (see Figure 6c). (4) Repeat step 3. successively for every level of H_Θ until all regions are connected. (5) Finally, close all neighborhoods on the negative z-axis according to H_Θ.

Obviously, with this construction all regions are internally connected. Furthermore all internally connected three-dimensional regions hold the same base relations as the two-dimensional realizations from which we started the construction. This is because all intervals on the z-axis are either contained in each other or are distinct, they have no common boundary points. All intervals corresponding to region X are contained in the intervals of region Y if and only if NTPP(X,Y) or TPP(X,Y). When two regions are disconnected they remain disconnected as they are not affiliated with the same neighborhoods. Two externally connected regions remain externally connected because every neighborhood was placed on the circle such that the common boundary points to its center. Therefore, if two of these regions are both affiliated with the same neighborhood, their pipes are externally connected and the horizontal connection of the single pipes is distinct. All other requirements of relations as, e.g., a common boundary point are already met by the pipes.

With the same construction, a $d+1$-dimensional realization of internally connected regions can be obtained from a d-dimensional realization of internally disconnected regions. All constructions kept the polytopic shape of the regions, so every region can be realized as a (d-dimensional) polytope. □

The restriction of regions to be polytopes can immediately be generalized to an arbitrary shape of regions.

6 APPLICABILITY OF THE CANONICAL MODEL

In the previous sections we reported about the existence of (reduced) RCC-8-models and how they can be mapped to topological spaces of different dimensions. In this section we study how RCC-8-models can be determined and how a realization can be generated from them. As there is a (reduced) RCC-8-model $\mathcal{M}$ for every consistent set of spatial relations Θ, and as it is always possible to generate a realization of $\mathcal{M}$, RCC-8 models are suitable for representing spatial regions with respect to their relations. RCC-8-models represent the characteristic points and information about their neighborhoods of a possible realization.

6.1 DETERMINATION OF RCC-8-MODELS

Given a set of spatial formulas Θ, we have to find a reduced RCC-8-model $\mathcal{M}$ for the modal encoding of RCC-8 such that only those worlds are boundary worlds of regions which are forced to be by the constraints. The Kripke frame of $\mathcal{M}$, i.e., the number of worlds and their accessibility relations are already known from the entailment constraints, but we have to find a valuation for every world and every region. For some worlds and some regions the valuation is already given from the constraints, for some it can be inferred using the constraints, for others it can be chosen. In order to make the inference step as easy as possible, we use the propositional encoding of RCC-8 with respect to a Kripke frame where every world w and every spatial variable X is transformed to a propositional atom X_w which is true if and only if X holds in w (Renz and Nebel 1997). The valuation of $\mathcal{M}$ can then be obtained from the satisfying assignment of the propositional formula. Even if the encoding of the reduced

RCC-8-models is not a Horn formula,[9] unit-resolution plus additional choices is sufficient for finding a satisfying assignment. As all clauses of the propositional encoding use worlds of the same RCC-8-cluster, the inference step is independent for every cluster. From Proposition 5.2 it is known which RCC-8-clusters contain a boundary world. Suppose that an RCC-8-cluster contains a boundary world, then the valuation of the two regions which introduced the boundary world can be chosen in all worlds of the RCC-8-cluster according to the relation of the two regions. The valuations of the other regions are either determined by unit-resolution or can be chosen according to their other valuations: If the valuation of a particular region in some world of the RCC-8-cluster is true, then the other valuations are also chosen as true, otherwise all valuations are chosen as false. If an RCC-8-cluster does not contain a boundary world, all worlds of the RCC-8-cluster have the same valuation. If the valuation of a region is not determined by unit-resolution it is chosen as false. With these choices a satisfying assignment is always found, even though the propositional formula is not Horn. As there are $O(n^2)$ worlds and n regions, there are $O(n^4)$ clauses (Renz and Nebel 1997), so a reduced RCC-8-model can be determined in time $O(n^4)$.

6.2 GENERATING A REALIZATION

Suppose we have given a reduced RCC-8-model of a consistent set of RCC-8 formulas Θ. We have to distinguish the tasks of generating a realization of internally connected and disconnected regions. A realization of disconnected regions in d-dimensional space can be obtained by placing the $O(n^2)$ different neighborhoods in the d-dimensional space and close each neighborhood as specified in Section 5. For this, the hierarchy H_Θ of regions must be known, which can be computed in time $O(n + P_\Theta)$ where $P_\Theta \in O(n^2)$ is the number of "part of" relations in Θ (see Footnote 8). Let $A_\Theta \in O(n)$ be the maximal number of regions affiliated with a neighborhood, then the closure of a neighborhood can be computed in time $O(A_\Theta)$.

Theorem 6.1 *Given a reduced* RCC-8 *model of a set of* RCC-8 *relations Θ, a realization in d-dimensional space ($d \geq 1$) can be generated in time $O(n^2 A_\Theta)$ when regions are allowed to be disconnected.*

In order to generate a realization of internally connected regions we can use the construction of the proof to Theorem 5.6. For every region we have to find the corresponding intervals on the z-axis. The number of intervals of a particular region X is equal to the number of regions with $H_\Theta = 1$ that are contained in X. Let $I_\Theta \in O(n)$ be the maximal number of regions with $H_\Theta = 1$ that are contained in a region.

Theorem 6.2 *Given a reduced* RCC-8 *model of a set of* RCC-8 *relations Θ, a realization of internally connected regions in d-dimensional space ($d \geq 3$) can be generated in time $O(n^2 A_\Theta I_\Theta)$.*

If $P_\Theta \in O(n^2)$ is the maximal number of neighborhoods affiliated with a region, every region can be realized as a polytope with $O(P_\Theta I_\Theta)$ vertices.

7 DISCUSSION & RELATED WORK

There is some work on identifying canonical models for the RCC axioms, i.e., determining what mathematical structures fulfill all the RCC axioms, as, e.g., every region has a non-tangential proper part (Randell *et al.* 1992). Gotts (1996) found that every connected and regular topological space is a model for the RCC axioms. Stell and Worboys (1997) identified a whole class of models base on Heyting structures. Both approaches only describe models for the RCC axioms, i.e., what kind of regions can be used at all. When additional constraints expressing relationships between regions are added, these results do not say anything about models anymore. [10] They are also by no means constructive, as they do not provide a way of effectively representing regions or generating realizations.

Previous approaches on dealing with dimension and internal connectedness of regions tried to specify predicates and suitable axioms in order to restrict dimension and connectedness of regions (Bennett 1996; Gotts 1994). As all regions must have the same dimension anyway, using our results it is not necessary to specify the dimension of regions explicitly if internally disconnected regions are permitted. If internally connected regions are required, these predicates only have an influence on the consistency of a set of spatial relations in one- or two-dimensional applications. In three- and higher-dimensional space all regions may be either internally connected or disconnected. Forcing internal connectedness of regions in two-dimensional space leads to difficult computational problems as there are no complete algorithms for dealing with this task. As Grigni et al. (1995) pointed out,

[9]This is because of the constraint $\Box(X \rightarrow \neg I \neg IX)$ which is transformed to $\bigwedge_{w \in W_0}(\neg X_w \vee X^1_w \vee X^2_w)$ in the notation of (Renz and Nebel 1997).

[10]Consider the hypothetical case of a set of spatial formulas which is realizable in two- but not in one-dimensional space. Then, a one-dimensional space is still a model of the RCC axioms, but not of RCC-8.

a well-known open problem in graph theory which is NP-hard but not known to be in NP (Kratochvíl 1991; Kratochvíl and Matoušek 1991) can be reduced to the consistency problem for two-dimensional internally connected regions.

It is certainly the better approach to have an additional connectedness predicate than forcing all regions to be internally connected which is done, e.g., by the similar calculus of Egenhofer (1991), as there are many applications where regions are in fact disconnected. Within the area of geographical information systems, e.g., which offer a great variety of possible applications, many countries or other geographical entities are not internally connected regions. In areas like computer vision it is often dealt with two-dimensional projections of the three-dimensional space where many connected objects are perceived as disconnected objects due to occlusion. In robot navigation, maps are often two-dimensional cuttings of a three-dimensional space.

With the result on realizations in one-dimensional space it becomes possible now to use RCC-8 for temporal reasoning tasks, in particular when non-convex intervals are allowed and the direction of the time is not important, as, e.g., in some scheduling problems. This is in contrast to previous approaches that used temporal calculi for spatial reasoning (Guesgen 1989).

8 SUMMARY

We identified a canonical model of RCC-8 based on Kripke semantics. In order to obtain a "topological" canonical model, we gave a topological interpretation of the Kripke models such that regions can be represented by points in the topological space and information about the neighborhood of these points with respect to the spatial relations holding between the regions. Using this canonical model, we proved that every consistent set of spatial formulas has a realization in any dimension when regions are not forced to be internally connected, which is the case for regions as used by RCC-8. Furthermore, we proved that for three- and higher dimensional space there is always a realization with internally connected regions. Further, we give for the first time algorithms for generating realizations of either internally connected or disconnected regions. Future work includes analyzing the usage of the canonical model for dealing with the special case of two-dimensional internally connected regions as well as analyzing the cognitive meaning of the canonical model (Knauff *et al.* 1997).

Acknowledgments

I would like to thank Bernhard Nebel and Tony Cohn for the fruitful discussions on the topic of the paper. This research has been supported by DFG as part of the project FAST-QUAL-SPACE, which is part of the DFG special research effort on "Spatial Cognition".

References

B. Bennett (1995). Modal logics for qualitative spatial reasoning. *Bulletin of the IGPL*, 4(1).

B. Bennett (1996). Carving up space: steps towards construction of an absolutely complete theory of spatial regions. In *Proc. JELIA'96*, LNCS 1126.

B.F. Chellas (1980). *Modal Logic: An Introduction.* Cambridge University Press, Cambridge, UK.

A.G. Cohn (1997). Qualitative spatial representation and reasoning techniques. In *Proc. KI-97*, LNCS 1303.

T.H. Cormen, C.E. Leiserson, and R.L. Rivest (1990). *Introduction to Algorithms.* MIT Press.

M.J. Egenhofer (1991). Reasoning about binary topological relations. In *Proc. SSD'91*, LNCS 525.

M.C. Fitting (1993). Basic modal logic. In *Handbook of Logic in Artificial Intelligence and Logic Programming - Vol. 1*, Oxford, Clarendon Press.

N.M. Gotts (1994). How far can we C? defining a 'doughnut' using connection alone. In *Proc. KR94*.

N.M. Gotts (1996). An axiomatic approach to topology for spatial information systems. Technical Report 96-25, Univ. of Leeds, School of Computer Studies.

M. Grigni, D. Papadias, and C. Papadimitriou (1995). Topological inference. In *Proc. IJCAI'95*.

H. Guesgen (1989). Spatial reasoning based on Allen's temporal logic. Technical Report TR-89-049, ICSI, Berkeley.

M. Knauff, R. Rauh, and J. Renz (1997). A cognitive assessment of topological spatial relations: Results from an empirical investigation. In *Proc. COSIT'97*, LNCS 1329.

J. Kratochvíl and J. Matoušek (1991). String graphs requiring exponential representations. *Journal of Combinatorial Theory, Series B*, 53:1–4.

J. Kratochvíl (1991). String graphs II: recognizing string graphs is NP-hard. *Journal of Combinatorial Theory, Series B*, 52:67–78.

O. Lemon (1996). Semantical foundations of spatial logics. In *Proc. KR'96*.

B. Nebel (1995). Computational properties of qualitative spatial reasoning: First results. In *Proc. KI-95*, LNCS 981.

D.A. Randell, Z. Cui, and A.G. Cohn (1992). A spatial logic based on regions and connection. In *Proc. KR'92*.

J. Renz and B. Nebel (1997). On the complexity of qualitative spatial reasoning: A maximal tractable fragment of the Region Connection Calculus. In *Proc. IJCAI'97*. Extended version under review for journal publication.

J. Stell and M. Worboys (1997). The algebraic structure of sets of regions. In *Proc. COSIT'97*, LNCS 1329.

Undecidability of Plane Polygonal Mereotopology

Christoph Dornheim[*]
Fachbereich Informatik
Universität Hamburg
Vogt-Kölln-Str. 30
D-22527 Hamburg, Germany

Abstract

This paper presents a mereotopological model of polygonal regions of the Euclidean plane and an undecidability proof of its first-order theory. Restricted to the primitive operations the model is a Boolean algebra. Its single primitive predicate defines simple polygons as the topologically simplest polygonal regions. It turns out that both the relations usually provided by mereotopologies and more subtle topological relations are elementarily definable in the model. Using these relations, Post's correspondence problem, known as undecidable, can be reduced to the decision problem of the model.

1 Introduction

Formalizing commonsense concepts of space has received much attention both in the philosophical literature and in recent AI research. Mereotopological theories as well as most calculi for spatial reasoning deal with spatial *regions*, i.e. the parts of space occupied by physical bodies, and their topological relations as intuitive concepts of our commonsense space. Whereas mereotopologies are designed to axiomatize the overall structure of regions in terms of topological relations, the research aim of *qualitative spatial reasoning* is the development of calculi to reason about regions and their relations as efficiently as possible (a recent survey is presented by Cohn [6]).

Mereotopologies are formal theories, mostly specified in predicate logic, that describe general properties of spatial regions in terms of topological and mereological relations. A relation is called topological if it is invariant under topological transformations, and mereological if it can be defined entirely by the part-whole relation. For an overview about mereology and mereotopology, especially the ways of defining mereotopology using mereological and topological primitives, we refer to [19] and [9]. The first approach in defining a mereotopology is due to Whitehead [20]; some other approaches are presented by Tarski [18], Randell et al. [16] and Borgo et al. [4].

Most mereotopologies are defined axiomatically by taking as axioms propositions that are considered intuitive for regions and their relations. The overall structure of these regions, together with their mereotopological relations, forms the intended model, which mereotopological theories are designed to axiomatize. Thus, to capture an intended model entirely, mereotopological theories are required to be complete. But the main difficulty in searching for a complete mereotopology is due to the fact that the intended model of regions is not described precisely as a particular model-theoretical structure. Rather than proving if an intuitively defined axiom system is complete it is a better method for such theoretical investigations to define a mereotopology semantically with respect to a particular model $\mathcal{M}$ representing the intended regions and relations. When using first-order logic this means that a mereotopology is defined as the elementary theory $Th(\mathcal{M})$ of $\mathcal{M}$, i.e. the set of all sentences of the elementary predicate language of $\mathcal{M}$ that hold in $\mathcal{M}$ (we use "*elementary*" as a synonym for "first-order"). Then, the problem in finding a complete axiomatic mereotopology is proving whether $Th(\mathcal{M})$ can be axiomatized elementarily. So far, such questions on semantics have gained only little attention in the literature; some models for mereotopology are proposed by Asher and Vieu [1] and by Pratt and Lemon [14].

These considerations are also important for spatial rea-

[*] The author's new address is: Institut für Informatik, Albert-Ludwigs-Universität Freiburg, Am Flughafen 17, D-79110 Freiburg, Germany, email: dornheim@informatik.uni-freiburg.de

soning. Since the underlying semantics of many calculi (e.g. [16], [2], [3]) are too weak to represent the intended regions, these calculi are of limited use, though they can be tractable [17]. Therefore, an appropriate mereotopological model $\mathcal{M}$ can serve equally as a formal semantics for calculi for spatial reasoning, because the statements on regions and relations, which are computed, can be expressed adequately as sentences of $Th(\mathcal{M})$. Thus, it is of considerable interest to know whether $Th(\mathcal{M})$ is decidable. For complete elementary theories like $Th(\mathcal{M})$ decidability is equivalent to axiomatizability (see theorem 3 in section 3), such that the decision problem of $Th(\mathcal{M})$ is relevant both for mereotopology and spatial reasoning.

In this paper we show that $Th(\mathcal{M})$ is already undecidable for the simple mereotopological model $\mathcal{M}$ of polygonal regions of the Euclidean plane. In section 2 we define $\mathcal{M}$ to consist of polygonal regions, some primitive mereological operations and a single primitive topological predicate, such that $\mathcal{M}$ turns out to be a topological Boolean algebra. Section 3 shows first that many topological relations are indeed definable in $\mathcal{M}$, which justifies $\mathcal{M}$ as a mereotopological model, and second, using these definability results, that $Th(\mathcal{M})$ is undecidable by reducing Post's correspondence problem to the decision problem of $Th(\mathcal{M})$. Finally, section 4 offers some conclusions.

2 The Model $\mathcal{M}$ of Plane Polygonal Regions

With the definition of the following mereotopological model $\mathcal{M}$ we propose a simple and plausible semantics for mereotopology of the plane. As usual, the Euclidean plane is represented by the two-dimensional real vector space $\mathbb{R}^2$, whose standard topology and geometry we use to define certain subsets of $\mathbb{R}^2$ as regions. We call $\mathcal{M}$ a *mereotopological model* if $\mathcal{M}$ satisfies the following conditions:

1. The elements of the domain M of $\mathcal{M}$ must be regular closed (or alternatively, regular open) to represent the intended regions, conceptualized as two-dimensional objects, adequately. As a consequence, they are all of the same dimensionality.

2. The primitive relations and functions of $\mathcal{M}$ must be topological or mereological. This ensures that $Th(\mathcal{M})$ contains only propositions about the mereotopological structure of M.

3. As we want $Th(\mathcal{M})$ to include all elementary sentences about the usually considered topological and mereological relations between regions, these relations, defined on M, must be elementarily definable in $\mathcal{M}$, i.e. expressible by formal relations in the predicate language induced by $\mathcal{M}$ (the precise definition is given in section 3).

4. All propositions of $Th(\mathcal{M})$ must hold for the intended regions and their relations.

In the choice of primitives for $\mathcal{M}$ we follow some approaches (e.g. [18], [9], [4], [14]) that distinguish clearly between mereological and topological notions by using primitives of both kinds. So, we take operations for building the sum, product and complement (with respect to $\mathbb{R}^2$) as mereological primitives and a single one-place relation as the topological primitive of $\mathcal{M}$. It is widely agreed that these operations for regions should form a Boolean algebra [19]. Although the null element is usually not allowed to represent a region due to philosophical considerations, we include it for technical convenience. It should be clear that this does not restrict the validity of the results.

Our definition is now as follows. Let x° be the *interior* and $\overline{x}$ the *closure* of $x \subseteq \mathbb{R}^2$. Then $x \subseteq \mathbb{R}^2$ is *regular closed* if $x = \overline{x^\circ}$. The set of all regular closed subsets of $\mathbb{R}^2$ is denoted by $RC(\mathbb{R}^2)$. The following operations are usually defined on $RC(\mathbb{R}^2)$ representing the mereological sum, product and complement, respectively:

$$x + y := x \cup y, \quad x \cdot y := \overline{(x \cap y)^\circ}, \quad -x := \overline{\mathbb{R}^2 \setminus x}.$$

As is well known, the structure $\mathcal{RC}(\mathbb{R}^2) := \langle RC(\mathbb{R}^2), +, \cdot, -, \emptyset, \mathbb{R}^2 \rangle$ is a Boolean algebra [12]. With regard to condition 1, we have to ensure that $\mathcal{M}$, restricted to these operations, turns out as a subalgebra of $\mathcal{RC}(\mathbb{R}^2)$. The elements of $RC(\mathbb{R}^2)$ themselves appear to be too general to represent the intended regions, e.g. $RC(\mathbb{R}^2)$ includes regions composed of infinitely many separate regions. Therefore we define the domain M of $\mathcal{M}$ to be a special proper subset of $RC(\mathbb{R}^2)$.

Let $a_1, \ldots, a_n \in \mathbb{R}^2$ and $[a_i, a_{i+1}]$ be the line segment between a_i and a_{i+1}. We call $[a_1, a_2] \cup \cdots \cup [a_{n-1}, a_n]$, denoted by $[a_1, \ldots, a_n]$, a *polygonal line.* If $a_1 = a_n$ the polygonal line is *closed.* If any point of $[a_1, \ldots, a_n] \backslash \{a_1, \ldots, a_n\}$ lies in exactly one line segment $[a_i, a_{i+1}]$ and any point of $\{a_1, \ldots, a_n\}$ belongs to at most two line segments, then we call the polygonal line *simple.* For simple closed polygonal lines the well-known *Jordan curve theorem* holds [13].

Theorem 1 *Let p be a simple closed polygonal line. Then $\mathbb{R}^2 \setminus p$ is a disjoint union of two open domains, a bounded* inner domain *D_p^I and an unbounded* outer domain *D_p^O, each of which has the boundary p. Moreover, the following properties are satisfied:*

1. *Any two points $a, b \in D_p^I$ are path-connected in D_p^I, i.e. there is a simple polygonal line $q \subset D_p^I$ with endpoints $a, b \in q$. The same holds for D_p^O.*
2. *Any simple polygonal line connecting a point of D_p^I with a point of D_p^O intersects p.* □

With the help of theorem 1 we first define the topologically simplest elements of M.

Definition 1 *Let p be a simple closed polygonal line. Then $\overline{D_p^I}$ is the* simple polygon *induced by p. With P we denote the set of all simple polygons.* □

Simple polygons are obviously in $RC(\mathbb{R}^2)$, such that the operations $+$, $\cdot$ and $-$ are declared for P.

Definition 2 *A* polygon *is a finite sum of simple polygons, i.e. $x_1 + \cdots + x_n$ with $n \geq 1$, $x_i \in P$. The set $M := \{\sum_{i=1}^{n} x_i, -(\sum_{i=1}^{n} x_i) \mid x_1, \ldots, x_n \in P\} \cup \{\emptyset, \mathbb{R}^2\}$ denotes the set of* polygonal regions. □

Using some general propositions on polygons [5] we can easily establish theorem 2.

Theorem 2 *$\langle M, +, \cdot, -, \emptyset, \mathbb{R}^2\rangle$ is a subalgebra of $\mathcal{RC}(\mathbb{R}^2)$.* □

This completes our construction of the mereological part of $\mathcal{M}$. Since topological relations are known not to be definable in a pure Boolean algebra, we have to include some topological primitive in $\mathcal{M}$. For this, we simply take the one-place relation P defined above, since P characterizes the simplest elements in M. Finally, our model $\mathcal{M}$ of polygonal regions is defined as:

Definition 3 $\mathcal{M} := \langle M, +, \cdot, -, \emptyset, \mathbb{R}^2, P\rangle$. □

3 Metamathematical Analysis of $\mathcal{M}$

In this section we first show by some definability results that $\mathcal{M}$ is indeed a mereotopological model (in particular with regard to condition 3). Using these results we are able to prove the undecidability of $Th(\mathcal{M})$. However, we first introduce the notions and notations of predicate logic [8] we need for analyzing $\mathcal{M}$.

A first-order predicate language L^S is determined by the set S of non-logical symbols. L^S contains all first-order expressions composed of the usual logical symbols and those of S. With L_0^S we denote the set of all *sentences* of L^S, i.e. expressions without any free variables. An *(elementary) theory* T is a consistent subset of L_0^S that is closed with respect to logical consequence. We say T is *(elementary) axiomatizable*, if there is a recursive $A \subseteq T$ from which any sentence of T follows; in this case, A is an *axiom system* of T. Moreover, T is called *complete*, if for any $\varphi \in L_0^S$ either $\varphi \in T$ or $\neg\varphi \in T$ holds. To say that $\varphi \in L_0^S$ is a logical consequence of A or is valid in a structure $\mathcal{S}$ we use the notation $A \models \varphi$ and $\mathcal{S} \models \varphi$ respectively. The next theorem [8] is fundamental for our purpose.

Theorem 3 *Let T be a theory. If T is axiomatizable, then it is recursively enumerable. If T is complete, then the following propositions are equivalent:*

1. *T is axiomatizable.*
2. *T is decidable.*
3. *T is recursively enumerable.* □

The set of non-logical symbols induced by $\mathcal{M}$ is $S_{\mathcal{M}} := \{+, \cdot, -, 0, 1, \mathsf{P}\}$; the symbols are to be interpreted with respect to $\mathcal{M}$ in the obvious manner, e.g. P denotes the relation P on M. Moreover, the *elementary theory of $\mathcal{M}$* is defined as $Th(\mathcal{M}) := \{\varphi \in L_0^{S_{\mathcal{M}}} \mid \mathcal{M} \models \varphi\}$, which is complete by definition.

We have to make precise what we mean by relations elementarily definable in $\mathcal{M}$. If $\varphi(\mathsf{x}_1, \ldots, \mathsf{x}_n) \in L^{S_{\mathcal{M}}}$ and $\mathsf{x}_1, \ldots, \mathsf{x}_n$ are its free variables then $\mathcal{M} \models \varphi[x_1, \ldots, x_n]$ means that φ is true when interpreted in $\mathcal{M}$ such that the variables x_i refer to $x_i \in M$ respectively. Then, $R \subseteq M^n$ is *elementarily definable in $\mathcal{M}$* if there is a $\varphi(\mathsf{x}_1, \ldots, \mathsf{x}_n) \in L^{S_{\mathcal{M}}}$ such that $(x_1, \ldots, x_n) \in M^n$ iff $\mathcal{M} \models \varphi[x_1, \ldots, x_n]$ for any $x_1, \ldots, x_n \in M$. Furthermore, if φ includes some defined relation symbols we write again $\mathcal{M} \models \varphi$ instead of extending $S_{\mathcal{M}}$ and $\mathcal{M}$. Since all defined relation symbols in φ can be replaced equivalently by its defining expressions of $L^{S_{\mathcal{M}}}$ this does not have any effect on definability or decidability questions.

3.1 Elementarily Definable Relations in $\mathcal{M}$

With the following propositions we present definability results[1] first for relations sufficient for defining easily the relations usually provided by mereotopological calculi. The other topological relations that are provably definable in $\mathcal{M}$ are more subtle and used in section 3.2 to encode Post's correspondence problem. In the remainder of this paper we shall abbreviate terms like $\mathsf{x} \cdot \mathsf{y}$ by xy.

Clearly, $\subseteq$ can be defined in $\mathcal{M}$ by the partial order $\leq$ usually associated with a Boolean algebra by

[1]The detailed proofs of the propositions given in this section are contained in [7].

$x \leq y \equiv_{def} x + y = y$. Also trivially, the set of polygons POL is definable in $\mathcal{M}$, since polygons are exactly the bounded non-null elements in M.

Proposition 1 *Let* $POL \subsetneq M$ *and* $\mathsf{POL}(x) \in L^{S\mathcal{M}}$ *be defined as:*

$$POL := \{x \in M \mid x = \sum_{i=1}^{n} x_i \text{ with } x_i \in P,\ n \geq 1\},$$

$$\mathsf{POL}(x) \equiv_{def} x \neq 0 \wedge \exists y(\mathsf{P}(y) \wedge x \leq y).$$

Then for all $x \in M$*:*

$$x \in POL \text{ iff } \mathcal{M} \models \mathsf{POL}[x]. \qquad \square$$

The relations defined in proposition 2, 3 and 4 respectively are crucial in defining topological relations. They distinguish fundamental cases of how the boundaries of two simple polygons x and y are related that share no common interior point. The first one, ECL, is the set of all pairs of simple polygons that are externally connected at a simple non-closed polygonal line. Second, ECP is the set of all pairs (x, y), such that x and y are externally connected at a single point. Figure 1 shows how $\mathsf{ECP}(x, y)$ defines ECP elementarily in $\mathcal{M}$. The set of all pairs of disconnected simple polygons is defined by DC.

Proposition 2 *Let* $ECL \subsetneq M^2$ *and* $\mathsf{ECL}(x, y) \in L^{S\mathcal{M}}$ *be defined as:*

$$ECL := \{(x, y) \in P^2 \mid x \cap y \text{ is a simple non-closed polygonal line } \},$$

$$\mathsf{ECL}(x, y) \equiv_{def} \mathsf{P}(x) \wedge \mathsf{P}(y) \wedge xy = 0 \wedge \mathsf{P}(x + y).$$

Then for all $x, y \in M$*:*

$$(x, y) \in ECL \text{ iff } \mathcal{M} \models \mathsf{ECL}[x, y]. \qquad \square$$

Proposition 3 *Let* $ECP \subsetneq M^2$ *and* $\mathsf{ECP}(x, y) \in L^{S\mathcal{M}}$ *be defined as:*

$$ECP := \{(x, y) \in P^2 \mid x \cap y = \{a\} \text{ for some } a \in \mathbb{R}^2\},$$

$$\begin{aligned}\mathsf{ECP}(x, y) \equiv_{def}\ & \\ & \mathsf{P}(x) \wedge \mathsf{P}(y) \wedge xy = 0 \wedge \neg\mathsf{P}(x + y) \wedge \\ & \exists v \exists w(\mathsf{P}(v) \wedge \mathsf{P}(w) \wedge \neg\mathsf{P}(v + w) \wedge \\ & \quad xv = 0 \wedge xw = 0 \wedge yv = 0 \wedge yw = 0 \wedge vw = 0 \wedge \\ & \quad \mathsf{P}(x + v) \wedge \mathsf{P}(x + w) \wedge \mathsf{P}(y + v) \wedge \mathsf{P}(y + w) \wedge \\ & \quad \mathsf{P}(x + v + y) \wedge \mathsf{P}(x + w + y) \wedge \mathsf{P}(x + v + w) \wedge \\ & \quad \mathsf{P}(y + v + w) \wedge \mathsf{P}(x + y + v + w)).\end{aligned}$$

Then for all $x, y \in M$*:*

$$(x, y) \in ECP \text{ iff } \mathcal{M} \models \mathsf{ECP}[x, y]. \qquad \square$$

Proposition 4 *Let* $DC \subsetneq M^2$ *and* $\mathsf{DC}(x, y) \in L^{S\mathcal{M}}$ *be defined as:*

$$DC := \{(x, y) \in P^2 \mid x \cap y = \emptyset\},$$

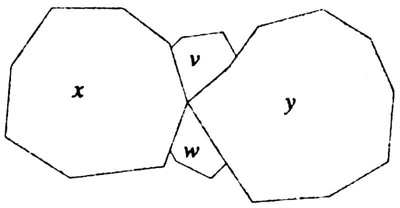


Figure 1: Simple polygons x, y, v, w used to define $\mathsf{ECP}(x, y)$ in proposition 3

$$\begin{aligned}\mathsf{DC}(x, y) \equiv_{def}\ & \mathsf{P}(x) \wedge \mathsf{P}(y) \wedge xy = 0 \wedge \\ & \neg(\exists v \exists w(v \leq x \wedge w \leq y \wedge \mathsf{ECP}(v, w))).\end{aligned}$$

Then for all $x, y \in M$*:*

$$(x, y) \in DC \text{ iff } \mathcal{M} \models \mathsf{DC}[x, y]. \qquad \square$$

It is obvious now that all the usual relations which are provided by mereotopology and spatial reasoning can be defined formally using ECL, ECP and DC. For instance, the connection relation C(x,y), often taken as the only primitive relation in mereotopological approaches such as the RCC-theory [16], can be defined by

$$\mathsf{C}(x, y) \equiv_{def} \exists v \exists w(v \leq x \wedge w \leq y \wedge \mathsf{ECP}(v, w)).$$

In accordance with the intended interpretation, $\mathcal{M} \models \mathsf{C}[x, y]$ holds for polygonal regions x and y iff x and y have a non-empty intersection. Another example of a RCC-relation that can be easily defined using ECP is TPP(x,y), the intended meaning of which is that x is a tangential proper part of y:

$$\begin{aligned}\mathsf{TPP}(x, y) \equiv_{def}\ & x \leq y \wedge x \neq y \wedge \\ & \exists v \exists w(v \leq x \wedge w \leq -y \wedge \mathsf{ECP}(v, w)).\end{aligned}$$

Due to these facts, $\mathcal{M}$ proves to be a mereotopological model, in particular with regard to condition 3 (see section 2). The next propositions deal with more special topological relations or general properties that hold for these relations. They are established with the purpose of preparing for the undecidability proof of $Th(\mathcal{M})$. Proposition 5 means that a simple polygon y, which is contained in the union of simple polygons $x_1, \ldots, x_n$ each two of which have at most one point in common, is already part of some x_i.

Proposition 5 *For all* $n \geq 1$ *let* $\sigma_n \in L_0^{S\mathcal{M}}$ *be*

$$\begin{aligned}\forall y \forall x_1 \cdots \forall x_n((\ & \bigwedge_{\substack{i,j \in \{1,\ldots,n\},\\ i \neq j}} (\mathsf{DC}(x_i, x_j) \vee \mathsf{ECP}(x_i, x_j)) \\ & \wedge \mathsf{P}(y) \wedge y \leq x_1 + \cdots + x_n) \\ & \Rightarrow (y \leq x_1 \vee \ldots \vee y \leq x_n)).\end{aligned}$$

Then for all $n \geq 1$*:* $\mathcal{M} \models \sigma_n$. $\square$

In definition 4 we introduce a relation that plays an important role not only for encoding Post's correspondence problem but generally in describing relevant relations between spatial objects.

Definition 4 *Let* $\mathsf{COM}(x,y) \in L^{S\mathcal{M}}$ *be defined as:*

$$\mathsf{COM}(x,y) \equiv_{def} \mathsf{P}(x) \land x \leq y \land \forall z((\mathsf{P}(z) \land x \leq z \land z \leq y) \Rightarrow x = z).$$

We call $x \in P$ *a* component *of* $y \in M$ *if* $\mathcal{M} \models \mathsf{COM}[x,y]$. □

Thus, a component x is a maximal simple polygon that is part of y. By definition of M, a polygonal region has at most finitely many different components. But it need not be the case, of course, that a polygonal region has any component at all (consider a simple polygon with a hole inside). The role of components comes up when considering a situation as described above: a union of simple polygons, each two of which have at most one point in common, can be separated entirely into its components, namely the simple polygons it is built of. This is what proposition 6 states, but since this separation property already follows from a simple subset $T'_{\mathcal{M}}$ of $Th(\mathcal{M})$ we formulate proposition 6 more generally.

Definition 5 *Let* $\Phi_{BA} \subset L_0^{\{+,\cdot,-,0,1\}}$ *denote the axiom system of ordinary Boolean algebra (see [12]). Then* $T'_{\mathcal{M}} \subset Th(\mathcal{M})$ *is defined as (for* $\sigma_n \in L_0^{S\mathcal{M}}$ *see proposition 5):*

$$T'_{\mathcal{M}} := \Phi_{BA} \cup \{\sigma_n \mid n \geq 1\} \cup \{\neg\mathsf{P}(0)\}.$$ □

Proposition 6 *For all* $n \geq 1$ *let* $\tau_n \in L_0^{S\mathcal{M}}$ *be*

$$\forall x_1 \cdots \forall x_n((\bigwedge_{\substack{i,j\in\{1,\ldots,n\},\\ i\neq j}} (\mathsf{DC}(x_i,x_j) \lor \mathsf{ECP}(x_i,x_j)) \land \mathsf{P}(x_1)) \Rightarrow (\bigwedge_{i\in\{1,\ldots,n\}} \mathsf{COM}(x_i, x_1 + \cdots + x_n) \land \forall y(\mathsf{COM}(y, x_1 + \cdots + x_n) \Rightarrow (y = x_1 \lor \cdots \lor y = x_n)))).$$

Then for all $n \geq 1$: $T'_{\mathcal{M}} \models \tau_n$. *In particular,* $\mathcal{M} \models \tau_n$. □

The next topological relation SDC is the set of all sums of disconnected simple polygons. In proposition 7 $x \in SDC$ is characterized formally as a polygon such that any simple polygon in x is contained in some component of x (μ_1) and, additionally, any distinct components of x are disconnected (μ_2).

Proposition 7 *Let* $SDC \subsetneq M$ *and* $\mathsf{SDC}(x) \in L^{S\mathcal{M}}$ *be defined as:*

$$SDC := \{x \in M \mid x = \sum_{i=1}^{n} x_i \text{ with } x_i \in P,\ n \geq 1,\ (x_i,x_j) \in DC \text{ if } i \neq j\},$$

$$\mathsf{SDC}(x) \equiv_{def} \mathsf{POL}(x) \land \mu_1(x) \land \mu_2(x)$$

with

$$\mu_1(x) \equiv_{def} \forall y((\mathsf{P}(y) \land y \leq x) \Rightarrow \exists z(y \leq z \land \mathsf{COM}(z,x))),$$

$$\mu_2(x) \equiv_{def} \forall v \forall w((\mathsf{COM}(v,x) \land \mathsf{COM}(w,x)) \Rightarrow (v = w \lor \mathsf{DC}(v,w))).$$

Then for all $x \in M$:

$$x \in SDC \text{ iff } \mathcal{M} \models \mathsf{SDC}[x].$$ □

The last relation SEQ defined in proposition 8 describes a sequence of simple polygons, as shown in figure 2. $(x_1, x) \in SEQ$ means that $x = x_1 + \cdots + x_n$ for some simple polygons $x_1, \ldots, x_n$, each two of which either intersect in a single point if their indices are neighbors or are disconnected otherwise. This condition allows x to be considered as a chain, because, by proposition 6, $x_1, \ldots, x_n$ are all the components of x, thus representing the distinct elements of the chain, whose endings are x_1 and x_n. If x_1 is fixed as the first element then x represents a sequence. In proposition 8 SEQ is elementarily defined by the expression SEQ requiring first that, similar to SDC, x is a polygon combined entirely of its components each two of which intersect at most in a single point and one of which is x_1, second that any partition of the components into v_1 and v_2, and hence x itself, is connected (ν_3), third that x_1 is externally connected to at most one component (ν_4), and finally that any component is externally connected to at most two distinct components (ν_5).

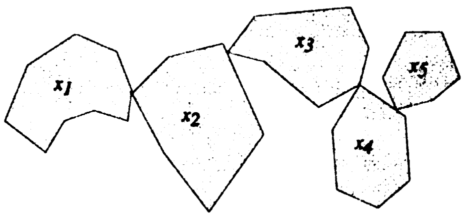


Figure 2: Representing a sequence by polygons: $(x_1, x_1 + \cdots, x_5) \in SEQ$

Proposition 8 *Let* $SEQ \subsetneq M^2$ *and* $\mathsf{SEQ}(x_1,x) \in L^{S\mathcal{M}}$ *be defined as:*

$$SEQ := \{(x_1,x) \in M^2 \mid x = \sum_{i=1}^{n} x_i \text{ with } x_i \in P,\ n \geq 1,\ (x_i,x_j) \in \begin{cases} ECP & \text{if } |i-j| = 1 \\ DC & \text{if } |i-j| > 1 \end{cases}\},$$

$$\mathsf{SEQ}(x_1,x) \equiv_{def} \mathsf{POL}(x) \land \mathsf{COM}(x_1,x) \land \nu_1(x) \land \nu_2(x) \land \nu_3(x) \land \nu_4(x_1,x) \land \nu_5(x)$$

with

$$\begin{aligned}
\nu_1(\mathsf{x}) &\equiv_{def} \forall \mathsf{y}((\mathsf{P}(\mathsf{y}) \wedge \mathsf{y} \leq \mathsf{x}) \Rightarrow \exists \mathsf{z}(\mathsf{y} \leq \mathsf{z} \wedge \mathsf{COM}(\mathsf{z},\mathsf{x}))),\\
\nu_2(\mathsf{x}) &\equiv_{def} \forall \mathsf{z}_1 \forall \mathsf{z}_2(\mathsf{COM}(\mathsf{z}_1,\mathsf{x}) \wedge \mathsf{COM}(\mathsf{z}_2,\mathsf{x})) \Rightarrow\\
&\qquad (\mathsf{z}_1 = \mathsf{z}_2 \vee \mathsf{ECP}(\mathsf{z}_1,\mathsf{z}_2) \vee \mathsf{DC}(\mathsf{z}_1,\mathsf{z}_2))),\\
\nu_3(\mathsf{x}) &\equiv_{def}\\
&\forall \mathsf{v}_1 \forall \mathsf{v}_2((\mathsf{v}_1 \neq 0 \wedge \mathsf{v}_2 \neq 0 \wedge \mathsf{v}_1\mathsf{v}_2 = 0 \wedge \mathsf{v}_1 + \mathsf{v}_2 = \mathsf{x} \wedge\\
&\qquad \forall \mathsf{z}(\mathsf{COM}(\mathsf{z},\mathsf{x}) \Rightarrow (\mathsf{z} \leq \mathsf{v}_1 \vee \mathsf{z} \leq \mathsf{v}_2))) \Rightarrow\\
&\qquad \exists \mathsf{z}_1 \exists \mathsf{z}_2(\mathsf{COM}(\mathsf{z}_1,\mathsf{x}) \wedge \mathsf{z}_1 \leq \mathsf{v}_1 \wedge\\
&\qquad\qquad \mathsf{COM}(\mathsf{z}_2,\mathsf{x}) \wedge \mathsf{z}_2 \leq \mathsf{v}_2 \wedge \mathsf{ECP}(\mathsf{z}_1,\mathsf{z}_2))),\\
\nu_4(\mathsf{x}_1,\mathsf{x}) &\equiv_{def} \forall \mathsf{z}_1 \forall \mathsf{z}_2((\mathsf{COM}(\mathsf{z}_1,\mathsf{x}) \wedge \mathsf{COM}(\mathsf{z}_2,\mathsf{x}) \wedge\\
&\qquad \mathsf{ECP}(\mathsf{z}_1,\mathsf{x}_1) \wedge \mathsf{ECP}(\mathsf{z}_2,\mathsf{x}_1))\\
&\qquad \Rightarrow \mathsf{z}_1 = \mathsf{z}_2),\\
\nu_5(\mathsf{x}) &\equiv_{def}\\
&\forall \mathsf{z} \forall \mathsf{z}_1 \forall \mathsf{z}_2 \forall \mathsf{z}_3((\mathsf{COM}(\mathsf{z},\mathsf{x}) \wedge \mathsf{COM}(\mathsf{z}_1,\mathsf{x}) \wedge \mathsf{COM}(\mathsf{z}_2,\mathsf{x})\\
&\qquad \wedge \mathsf{COM}(\mathsf{z}_3,\mathsf{x}) \wedge \mathsf{ECP}(\mathsf{z}_1,\mathsf{z}) \wedge \mathsf{ECP}(\mathsf{z}_2,\mathsf{z})\\
&\qquad \wedge \mathsf{ECP}(\mathsf{z}_3,\mathsf{z}))\\
&\qquad \Rightarrow (\mathsf{z}_1 = \mathsf{z}_2 \vee \mathsf{z}_1 = \mathsf{z}_3 \vee \mathsf{z}_2 = \mathsf{z}_3)).
\end{aligned}$$

Then for all $x_1, x \in M$:

$$(x_1, x) \in SEQ \quad \textit{iff} \quad \mathcal{M} \models \mathsf{SEQ}[x_1, x]. \qquad \square$$

To complete the preparation of the undecidability proof presented in the next section proposition 9 ensures the existence of polygons in M that enable the encoding of Post's correspondence problem. For any n there are simple polygons x_{ij}, $i, j \in \{1, \ldots, n\}$, that are arranged like squares of an $n \times n$ chessboard, as shown in figure 3. Then, the sentence ξ_n in proposition 9 guarantees some topological relations to hold between these x_{ij} as well as between special sums $\Phi_{i,j,k,l}$ of them, which are determined entirely by x_{ij} and x_{kl}.

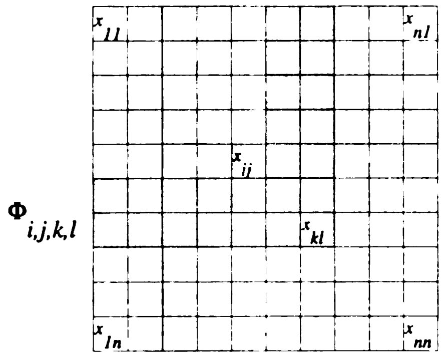


Figure 3: Simple polygons composed by squares of a chessboard

Proposition 9 *For all $n \geq 1$ let $\xi_n \in L_0^{S_\mathcal{M}}$ be*

$$\begin{aligned}
&\exists \mathsf{x}_{11} \cdots \exists \mathsf{x}_{n1} \exists \mathsf{x}_{12} \cdots \exists \mathsf{x}_{n2} \cdots \exists \mathsf{x}_{1n} \cdots \exists \mathsf{x}_{nn}\\
&\quad (\bigwedge_{i \neq k \vee j \neq l} \mathsf{x}_{ij}\mathsf{x}_{kl} = 0 \ \wedge \bigwedge_{i,j} \mathsf{P}(\mathsf{x}_{ij}) \ \wedge \bigwedge_{\substack{|i-k| \geq 2\\ \vee |j-l| \geq 2}} \mathsf{DC}(\mathsf{x}_{ij}, \mathsf{x}_{kl}) \ \wedge\\
&\quad \bigwedge_{\substack{i<k,\\ j<l}} \mathsf{P}(\Phi_{i,j,k,l}) \wedge \bigwedge_{\substack{i<k \wedge j<l,\\ k+1<k' \wedge l+1<l'}} \mathsf{ECP}(\Phi_{i,j,k,l}, \Phi_{k+1,l+1,k',l'})\\
&\quad \wedge \bigwedge_{\substack{i<k \wedge j<l,\\ i'<k' \wedge j'<l',\\ k+1<i' \wedge l+1<j'}} \mathsf{DC}(\Phi_{i,j,k,l}, \Phi_{i',j',k',l'}) \ )
\end{aligned}$$

with $\Phi_{i,j,k,l} :=$

$$\begin{aligned}
&\mathsf{x}_{i,j} + (\mathsf{x}_{i+1,1} + \cdots + \mathsf{x}_{k,1}) + \cdots + (\mathsf{x}_{i+1,j} + \cdots + \mathsf{x}_{k,j})\\
&\quad + (\mathsf{x}_{1,j+1} + \cdots + \mathsf{x}_{k,j+1}) + \cdots + (\mathsf{x}_{1,l} + \cdots + \mathsf{x}_{k,l}).
\end{aligned}$$

Then for all $n \geq 1$: $\mathcal{M} \models \xi_n$. $\square$

3.2 Undecidability of $Th(\mathcal{M})$

Using the propositions of the last section we are now able to show $Th(\mathcal{M})$ to be undecidable. Actually, we prove a slightly stronger result. We define an axiomatic fragment $T_\mathcal{M}$ of $Th(\mathcal{M})$ to consist of all properties of $\mathcal{M}$ that are sufficient for encoding Post's correspondence problem. As we will see in theorem 4, any set T of elementary sentences which is valid in $\mathcal{M}$ and includes $T_\mathcal{M}$ turns out to be undecidable.

Definition 6 *Let $T_\mathcal{M} \subset Th(\mathcal{M})$ be defined as*

$$\begin{aligned}
T_\mathcal{M} &:= T'_\mathcal{M} \cup \{\xi_n \mid n \geq 1\}\\
&= \Phi_{BA} \cup \{\neg \mathsf{P}(0)\} \cup \{\sigma_n,\ \xi_n \mid n \geq 1\}. \qquad \square
\end{aligned}$$

Post's correspondence problem (PCP) (restricted to the alphabet $\{0,1\}$) is defined as follows. A *PCP*-instance is given by a list $(\alpha_1, \beta_1), \ldots, (\alpha_m, \beta_m)$ of pairs of non-empty strings over $\{0,1\}$. This instance has a solution if there is any sequence of integers $i_1, \ldots, i_k$ with $1 \leq i_j \leq m$, such that $\alpha_{i_1} \cdots \alpha_{i_k} = \beta_{i_1} \cdots \beta_{i_k}$. Post's correspondence problem is then to determine if a given *PCP*-instance has a solution. As it is well-known, this problem is undecidable [11].

The key in proving any $T \subset L_0^{S_\mathcal{M}}$ with $T_\mathcal{M} \subseteq T \subseteq Th(\mathcal{M})$ to be undecidable is to reduce Post's correspondence problem to the decision problem of T, that is, to associate with any *PCP*-instance C a sentence $\Psi_C \in L_0^{S_\mathcal{M}}$, such that $T \models \Psi_C$ iff C has a solution. The sentence Ψ_C is given in definition 7 using some operations on strings. For any string $\delta \in \{0,1\}^+$ $|\delta|$ denotes the length of δ, $\delta[i]$ denotes the ith symbol in δ, if $1 \leq i \leq |\delta|$, and $\delta[i,j]$ denotes the substring $\delta[i]\ \delta[i+1] \cdots \delta[j]$, if $1 \leq i \leq j \leq |\delta|$.

Definition 7 *Let $\Delta_\delta(\mathsf{x},\mathsf{c},\mathsf{a},\mathsf{a}_1,\mathsf{z}) \in L^{S_\mathcal{M}}$ for any $\delta \in \{0,1\}^+$ be defined as*

$\Delta_\delta(\mathsf{x},\mathsf{c},\mathsf{a},\mathsf{a}_1,\mathsf{z}) \equiv_{def}$

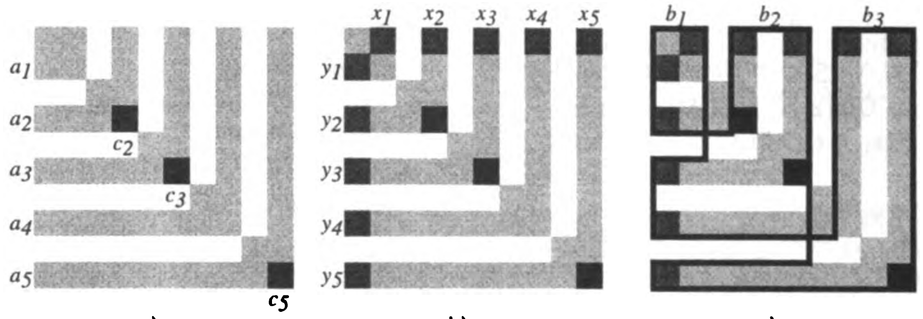


Figure 4: Encoding a solution of a *PCP*-instance by polygons: in a) the a_i are the light-grey colored and the c_l are the black colored simple polygons contained in a_l, in b) the x_i and y_i respectively are the dark-grey colored simple polygons, and in c) the b_j are the simple polygons marked by their borderlines.

$$\begin{aligned}
&\exists v_1 \cdots \exists v_{|\delta|}\\
&(\mathsf{COM}(v_1,a) \wedge \cdots \wedge \mathsf{COM}(v_{|\delta|},a) \wedge\\
&\mathsf{ECP}(v_1,v_2) \wedge \mathsf{ECP}(v_2,v_3) \wedge \cdots \wedge \mathsf{ECP}(v_{|\delta|-1},v_{|\delta|})\\
&\wedge \bigwedge_{i\neq j} v_i \neq v_j \wedge\\
&\forall s((s \leq a \wedge \forall d(\mathsf{COM}(d,a) \Rightarrow (d \leq s \vee ds = 0)) \wedge\\
&\quad \mathsf{SEQ}(a_1,s) \wedge v_{|\delta|} \leq s) \Rightarrow v_1 \leq s) \wedge\\
&zx = (v_1 + \cdots + v_{|\delta|})x \wedge\\
&\Omega_{\delta[1]}(v_1 c) \wedge \cdots \wedge \Omega_{\delta[|\delta|]}(v_{|\delta|}c))
\end{aligned}$$

$$\textit{with } \Omega_{\delta[i]}(v_i c) = \begin{cases} v_i c \neq 0 & \textit{if } \delta[i] = 1\\ v_i c = 0 & \textit{if } \delta[i] = 0. \end{cases}$$

Let $C = \{(\alpha_1,\beta_1),\ldots,(\alpha_m,\beta_m)\}$, *for* $\alpha_j,\beta_j \in \{0,1\}^+$, *be an arbitrary* PCP-*instance. Then,* $\Psi_C \in L_0^{\mathcal{SM}}$ *is defined as*

$$\begin{aligned}
&\Psi_C \equiv_{def}\\
&\quad \exists x \exists y \exists a \exists a_1 \exists b \exists b_1 \exists c\\
&\quad (\mathsf{SEQ}(a_1,a) \wedge \mathsf{SEQ}(b_1,b) \wedge \mathsf{SDC}(x) \wedge \mathsf{SDC}(y) \wedge\\
&\quad x \leq a \wedge y \leq a \wedge x \leq b \wedge y \leq b \wedge a_1 \leq b_1 \wedge\\
&\quad \lambda_1(a,x,y) \wedge \lambda_2(a,b,x,y) \wedge \lambda_3(x,y,a,a_1,b,c))
\end{aligned}$$

with

$$\begin{aligned}
&\lambda_1(a,x,y) \equiv_{def}\\
&\quad \forall z(\mathsf{COM}(z,a) \Rightarrow (\mathsf{COM}(zx,x) \wedge \mathsf{COM}(zy,y))),\\
&\lambda_2(a,b,x,y) \equiv_{def}\\
&\quad \forall z_1 \forall z_2((\mathsf{COM}(z_1,b) \wedge \mathsf{COM}(z_2,b) \wedge \mathsf{ECP}(z_1,z_2)) \Rightarrow\\
&\qquad (\exists v_1 \exists v_2(\mathsf{COM}(v_1,a) \wedge \mathsf{COM}(v_2,a) \wedge v_1 x \leq z_1\\
&\qquad\quad \wedge v_2 x \leq z_2 \wedge \mathsf{ECP}(v_1,v_2)) \wedge\\
&\qquad \exists u_1 \exists u_2(\mathsf{COM}(u_1,a) \wedge \mathsf{COM}(u_2,a) \wedge u_1 y \leq z_1\\
&\qquad\quad \wedge u_2 y \leq z_2 \wedge \mathsf{ECP}(u_1,u_2)))),\\
&\lambda_3(x,y,a,a_1,b,c) \equiv_{def}\\
&\quad \forall z(\mathsf{COM}(z,b) \Rightarrow\\
&\qquad (\bigvee_{j=1,\ldots,m} (\Delta_{\alpha_j}(x,c,a,a_1,z) \wedge \Delta_{\beta_j}(y,c,a,a_1,z)))).
\end{aligned}$$

□

To illustrate how Ψ_C encodes in $\mathcal{M}$ the solvability of a *PCP*-instance C, consider the case where C is given by $\alpha_1 = 0$, $\alpha_2 = 11$, $\alpha_3 = 01$ and $\beta_1 = 01$, $\beta_2 = 10$, $\beta_3 = 1$. Obviously, $(1,2,3)$ is a solution of C, since $\alpha_1\ \alpha_2\ \alpha_3 = 01101 = \beta_1\ \beta_2\ \beta_3$. The fact that this is a solution of C can be represented by polygons satisfying certain topological relations, as shown in figure 4. The string 01101 induced by the solution is represented by polygon $a = a_1 + \cdots + a_5$ together with polygon $c = c_2 + c_3 + c_5$, shown in a), because a represents a sequence whose ith element, represented by a_i, is "1" if $c_i \subset a_i$, otherwise "0". Since the polygons x_i and y_i respectively are in a one-to-one correspondence to the a_i, as shown in b) by $x_i, y_i \subset a_i$, the string 01101 is also represented both by polygon $x = x_1 + \cdots + x_5$ and by polygon $y = y_1 + \cdots + y_5$. This is what $\mathcal{M} \models \lambda_1[a,x,y]$ means.

Figure c) includes all polygons that are involved in encoding that $(1,2,3)$ is a solution of C. This is due to the fact that each b_j represents the jth index of the solution: b_j includes exactly the parts of x and y which represent the corresponding strings α_{i_j} and β_{i_j} respectively. More formally, this can be expressed by $\mathcal{M} \models \lambda_3[x,y,a,a_1,b,c]$, where b is $b_1 + b_2 + b_3$. For instance, we have $\mathcal{M} \models \Delta_{\alpha_3}[x,c,a,a_1,b_3]$, i.e. there are distinct components v_1 and v_2 of a which represent (by $(v_1,v_2) \in ECP$ and by the formula containing the variable s) a subsequence of a of the same order. Since $v_1 + v_2$ and b_3 must contain the same components x_i, we get $v_1 = a_4$ and $v_2 = a_5$. Thus $v_1 + v_2$ represents the string α_3.

Finally, $\mathcal{M} \models \lambda_2[a,b,x,y]$ ensures that the concatenation of the corresponding strings, i.e. $\alpha_1\ \alpha_2\ \alpha_3$ and $\beta_1\ \beta_2\ \beta_3$, results in the same string, namely 01101. Hence, $b = b_1 + b_2 + b_3$ encodes the whole solution $(1,2,3)$.

The polygons a, a_1, b, b_1, x, y, c satisfy all properties Ψ_C requires to hold, thus $\mathcal{M} \models \Psi_C$. In the following lemma 1 we show in detail how to construct a particular solution of a given *PCP*-instance by such polygons.

Lemma 1 *Let* $C = \{(\alpha_1, \beta_1), \ldots, (\alpha_m, \beta_m)\}$ *be a* PCP*-instance. Then* C *has a solution if* $\mathcal{M} \models \Psi_C$.

Proof: Suppose $\mathcal{M} \models \Psi_C$. By propositions 7 and 8 it follows that some polygonal regions $x, y, a, a_1, b, b_1, c \in M$ exist satisfying $(a_1, a), (b_1, b) \in SEQ$ and $x, y \in SDC$. Thus, $a_1 \in P$, $a = a_1 + \cdots + a_{n_a}$, $n_a \geq 1$ for some $a_2, \ldots, a_{n_a} \in P$, and $b_1 \in P$, $b = b_1 + \cdots + b_k$, $k \geq 1$ for some $b_2, \ldots, b_k \in P$. a_i, a_j with $i, j \in \{1, \ldots, n_a\}$ and b_i, b_j with $i, j \in \{1, \ldots, k\}$ respectively satisfy the condition

$$(a_i, a_j),\ (b_i, b_j) \in \begin{cases} ECP & \text{if} \quad |i-j| = 1 \\ DC & \text{if} \quad |i-j| > 1. \end{cases} \qquad (1)$$

In particular, $a_i a_j = \emptyset$ and $b_i b_j = \emptyset$ if $i \neq j$. From proposition 6 we obtain that the a_i are the only components of a, that is $(z, a) \in COM$ holds exactly for every $z \in \{a_1, \ldots, a_{n_a}\}$. Equally, $b_1, \ldots, b_k$ are exactly the components of b.

Furthermore, $x = x_1 + \cdots + x_{n_x}$, $n_x \geq 1$ for some $x_1, \ldots, x_{n_x} \in P$ and $y = y_1 + \cdots + y_{n_y}$, $n_y \geq 1$ for some $y_1, \ldots, y_{n_y} \in P$. x_i, x_j with $i, j \in \{1, \ldots, n_x\}$ and y_i, y_j with $i, j \in \{1, \ldots, n_y\}$ respectively satisfy the condition

$$(x_i, x_j),\ (y_i, y_j) \in DC \text{ if } i \neq j \qquad (2)$$

with the consequence that $x_i x_j = \emptyset$ and $y_i y_j = \emptyset$ if $i \neq j$. Again by proposition 6, x_i and y_i are the only components of x and y, respectively. In the following steps we construct a solution of C.

1) We show the functions F_x and F_y defined as

$$F_x : \{a_1, \ldots a_{n_a}\} \longrightarrow \{x_1, \ldots, x_{n_x}\},\ F_x(a_i) := a_i x,$$
$$F_y : \{a_1, \ldots a_{n_a}\} \longrightarrow \{y_1, \ldots, y_{n_y}\},\ \ F_y(a_i) := a_i y$$

to be one-to-one mappings. For the remainder of the proof we then define $n := n_x = n_a = n_y$.

This follows from $\mathcal{M} \models \lambda_1[a, x, y]$. For any a_j there exists exactly one x_i such that $a_j x = x_i$. Thus, F_x is indeed an overall defined function which maps different a_j to different x_i. From $x \subseteq a$ and proposition 5 it further follows for any x_i that $x_i \subseteq a_j$ holds for some a_j. This implies $x_i = a_j x = F_x(a_j)$. So F_x, and analogously F_y, turn out to be one-to-one functions.

2) We define the string $\pi \in \{0,1\}^+$ with length $|\pi| = n$ by

$$\pi[i] := \begin{cases} 1 & \text{if} \quad a_i c \neq \emptyset \\ 0 & \text{if} \quad a_i c = \emptyset. \end{cases}$$

π is the string that is induced by the special solution of C we construct in the following.

3) Let $G_x, H_x, G_y, H_y : \{1, \ldots, k\} \longrightarrow \{1, \ldots, n\}$ be defined as:

$$\begin{aligned} G_x(i) &:= \min\{j \mid F_x(a_j) \subseteq b_i\}, \\ H_x(i) &:= \max\{j \mid F_x(a_j) \subseteq b_i\}, \\ G_y(i) &:= \min\{j \mid F_y(a_j) \subseteq b_i\}, \\ H_y(i) &:= \max\{j \mid F_y(a_j) \subseteq b_i\}. \end{aligned}$$

We show that for all $i \in \{1, \ldots, k\}$, $j \in \{1, \ldots, n\}$:

a) $G_x(i) \leq j \leq H_x(i)$ iff $F_x(a_j) \subseteq b_i$.

b) $G_y(i) \leq j \leq H_y(i)$ iff $F_y(a_j) \subseteq b_i$.

c) $\pi[G_x(i), H_x(i)] = \alpha_{l(i)}$, $\pi[G_y(i), H_y(i)] = \beta_{l(i)}$ for some $l(i) \in \{1, \ldots, m\}$.

This can be proven from $\mathcal{M} \models \lambda_3[x, y, a, a_1, b, c]$. For any b_i there exists some $j \in \{1, \ldots, m\}$ satisfying

$$\begin{aligned} &\mathcal{M} \models \Delta_{\alpha_j}[x, c, a, a_1, b_i], \\ &\mathcal{M} \models \Delta_{\beta_j}[y, c, a, a_1, b_i]. \end{aligned} \qquad (3)$$

In the following we consider only (3). The $v_1, \ldots, v_{|\alpha_j|} \in M$ are pairwise distinct elements from $\{a_1, \ldots, a_n\}$. Suppose $v_{|\alpha_j|} = a_r$. Then we define $s := a_1 + \cdots + a_r$. From (3) we can conclude that $v_1 \subseteq s$.

In case of $|\alpha_j| > 1$ we get by (3) $(v_{|\alpha_j|-1}, v_{|\alpha_j|}) \in ECP$, thus by (1) either $v_{|\alpha_j|-1} = a_{r+1}$ or $v_{|\alpha_j|-1} = a_{r-1}$. Suppose $v_{|\alpha_j|-1} = a_{r+1}$. From $(v_{|\alpha_j|-2}, v_{|\alpha_j|-1}) \in ECP$ and $v_{|\alpha_j|-2} \neq v_{|\alpha_j|}$ it then follows $v_{|\alpha_j|-2} = a_{r+2}$. Concluding in the same way leads to $v_1 = a_{r+|\alpha_j|-1}$, which contradicts $v_1 \subseteq s$. Therefore we obtain

$$v_{|\alpha_j|} = a_r,\ v_{|\alpha_j|-1} = a_{r-1}, \ldots,\ v_1 = a_{r-|\alpha_j|+1}.$$

Let $q := r - |\alpha_j| + 1$ such that $1 \leq q \leq r \leq n$. Then we get by (3)

$$b_i x = (a_q + \cdots + a_r)x = F_x(a_q) + \cdots + F_x(a_r). \quad (4)$$

Thus, $F_x(a_e) \subseteq b_i$ for any $q \in \{q, \ldots, r\}$. Conversely, suppose $F_x(a_e) \subseteq b_i$ for some $e \in \{1, \ldots, n\}$. Since $F_x(a_e) \subseteq x$ it follows from (4) that $F_x(a_e) = F_x(a_p)$ for some p with $q \leq p \leq r$. Because F_x is one-to-one we get $e = p$. This means in fact that $G_x(i) = q$, $H_x(i) = r$ and proposition a) holds.

Moreover, it follows from (3) that

$$a_{q+e-1} c = v_e c \begin{cases} \neq \emptyset & \text{if} \quad \alpha_j[e] = 1 \\ = \emptyset & \text{if} \quad \alpha_j[e] = 0 \end{cases}$$

and thus, by definition of π, $\pi[q + e - 1] = \alpha_j[e]$ for any $e \in \{1, \ldots, |\alpha_j|\}$. This means

$$\pi[q, r] = \pi[q] \cdots \pi[r] = \alpha_j[1] \cdots \alpha_j[|\alpha_j|] = \alpha_j.$$

Similarly we can show by $\mathcal{M} \models \Delta_{\beta_j}[y, c, a, a_1, b_i]$ that proposition b) and $\pi[G_y(i), H_y(i)] = \beta_j$ must hold. Hence, proposition c) holds with $l(i) := j$.

4) We show that

d) $G_x(1) = 1,\ H_x(k) = n,$
$G_x(i) = H_x(i-1) + 1$ for $i \in \{2, \dots, k\}$

e) $G_y(1) = 1,\ H_y(k) = n,$
$G_y(i) = H_y(i-1) + 1$ for $i \in \{2, \dots, k\}$.

To show d) we first note that

$$\{1, \dots, n\} = \dot{\bigcup_{i=1,\dots,k}} \{G_x(i), \dots, H_x(i)\}. \qquad (5)$$

This holds since by a) each two sets of the right-hand side do not intersect. Conversely, we have for any $i \in \{1, \dots, n\}$ $F_x(a_i) \subseteq x \subseteq b$ and thus, by proposition 5, $F_x(a_i) \subseteq b_j$ for some $j \in \{1, \dots, k\}$. With a) this means $i \in \{G_x(j), \dots, H_x(j)\}$.

Since $a_1 \subseteq b_1$ we get by a) $G_x(1) \leq 1 \leq H_x(1)$ and thus $G_x(1) = 1$. Assuming $k > 1$ we show $G_x(i) = H_x(i-1) + 1$ for $i \in \{2, \dots, k\}$ by induction using $\mathcal{M} \models \lambda_2[a, b, x, y]$.

Suppose $i = 2$. From $\mathcal{M} \models \lambda_2[a, b, x, y]$ and (1) it follows that $a_s x \subseteq b_1$ and $a_t x \subseteq b_2$ for some components a_s, a_t of a satisfying $(a_s, a_t) \in ECP$, i.e. $|s - t| = 1$. Then, by a),

$$1 = G_x(1) \leq s \leq H_x(1) \text{ and } G_x(2) \leq t \leq H_x(2)$$

from which $s = H_x(1)$ and $t = s + 1 = G_x(2)$ follows by (5), that is $G_x(2) = H_x(1) + 1$.

Suppose $i \in \{2, \dots, k-1\}$ and $G_x(j) = H_x(j-1) + 1$ to hold for all j with $2 \leq j \leq i$. Again, we have $a_s x \subseteq b_i$ and $a_t x \subseteq b_{i+1}$ for some components a_s, a_t of a satisfying $(a_s, a_t) \in ECP$, i.e. $|s - t| = 1$, and

$$G_x(i) \leq s \leq H_x(i) \text{ and } G_x(i+1) \leq t \leq H_x(i+1).$$

Thus, by (5), either $s = H_x(i), t = s+1 = G_x(i+1)$ or $s = G_x(i), t = s-1 = H_x(i+1)$. Since from the latter we can conclude $H_x(i+1) = G_x(i) - 1 = H_x(i-1)$, which contradicts (5), $G_x(i+1) = H_x(i) + 1$ must necessarily hold.

Finally, we get $H_x(k) = n$ from (5) and

$$\bigcup_{i=1,\dots,k-1} \{G_x(i), \dots, H_x(i)\} = \{1, \dots, H_x(k-1)\}.$$

The proof of e) is analogous.

5) Using propositions c), d) and e) we can now easily show that $(l(1), \dots, l(k))$ is a solution of C:

$$\begin{gathered} \alpha_{l(1)}\ \alpha_{l(2)} \cdots \alpha_{l(k)} = \\ \pi[G_x(1), H_x(1)]\ \pi[G_x(2), H_x(2)] \cdots \pi[G_x(k), H_x(k)] \\ = \pi = \\ \pi[G_y(1), H_y(1)]\ \pi[G_y(2), H_y(2)] \cdots \pi[G_y(k), H_y(k)] \\ = \beta_{l(1)}\ \beta_{l(2)} \cdots \beta_{l(k)}. \end{gathered}$$

□

Conversely, lemma 2 shows that Ψ_C is already a consequence of $T_{\mathcal{M}}$ provided that C has a solution.

Lemma 2 *Let* $C = \{(\alpha_1, \beta_1), \dots, (\alpha_m, \beta_m)\}$ *be a solvable* PCP-*instance. Then,* $T_{\mathcal{M}} \models \Psi_C$ *holds.*

Proof: Let $(i_1, \dots, i_k)$ be a solution of C. Let $\mathcal{A} = \langle A, +^A, \cdot^A, -^A, 0^A, 1^A, P^A \rangle$ be an arbitrary model of $T_{\mathcal{M}}$. Then we have to show $\mathcal{A} \models \Psi_C$. For this, let $\leq^A$, POL^A, ECP^A, DC^A, COM^A, SDC^A and SEQ^A be the relations defined on A by $\leq$, POL, ECP, DC, COM, SDC and SEQ, respectively, such that $x \leq^A y$ iff $\mathcal{A} \models \leq[x, y]$ for any $x, y \in A$, etc.

We define $\pi := \alpha_{i_1} \cdots \alpha_{i_k} = \beta_{i_1} \cdots \beta_{i_k} \in \{0,1\}^+$ and $n := |\pi|$. From $\xi_{2n} \in T_{\mathcal{M}}$ it follows that there exist elements $d_{i,j} \in A$, $i, j \in \{1, \dots, 2n\}$, satisfying the following properties:

$$\begin{aligned} & d_{i,j} \cdot^A d_{k,l} = 0^A \text{ if } (i,j) \neq (k,l), \\ & d_{i,j} \in P^A, \\ & (d_{i,j}, d_{k,l}) \in DC^A \text{ if } |i-k| \geq 2 \text{ or } |j-l| \geq 2, \\ & \Phi_{i,j,k,l} \in P^A, \\ & (\Phi_{i,j,k,l}, \Phi_{k+1,l+1,k',l'}) \in ECP^A, \\ & (\Phi_{i,j,k,l}, \Phi_{i',j',k',l'}) \in DC^A, \text{ if } k+1 < i',\ l+1 < j' \end{aligned}$$

where $\Phi_{i,j,k,l}$ is defined, if $i < k$ and $j < l$, as

$$\begin{aligned} \Phi_{i,j,k,l} := {} & d_{i,j} +^A \textstyle\sum^A_{r=1,\dots,j} (d_{i+1,r} +^A \cdots +^A d_{k,r}) \\ & +^A \textstyle\sum^A_{r=j+1,\dots,l} (d_{1,r} +^A \cdots +^A d_{k,r}). \end{aligned}$$

Next we define elements $a, a_1, c, x, y, b, b_1 \in A$ that validate Ψ_C in $\mathcal{A}$:

1. $a := a_1 +^A \cdots +^A a_n$ with $a_i := \Phi_{2i-1,2i-1,2i,2i}$
2. $c := c_1 +^A \cdots +^A c_n$ with
$$c_i := \begin{cases} d_{2i,2i} & \text{if } \pi[i] = 1 \\ 0^A & \text{if } \pi[i] = 0 \end{cases}$$
3. $x := x_1 +^A \cdots +^A x_n$ with $x_i := d_{2i,1}$
4. $y := y_1 +^A \cdots +^A y_n$ with $y_i := d_{1,2i}$

5. $b := b_1 +^A \cdots +^A b_k$ with $b_j := \Phi_{e,f,g,h}$ where

$$\begin{aligned} e &= 2|\alpha_{i_1}\cdots\alpha_{i_{j-1}}|+1, \\ f &= 2|\beta_{i_1}\cdots\beta_{i_{j-1}}|+1, \\ g &= 2|\alpha_{i_1}\cdots\alpha_{i_j}|, \\ h &= 2|\beta_{i_1}\cdots\beta_{i_j}|. \end{aligned}$$

1) We show that $(a_1, a) \in SEQ^A$. By definition of the a_i we get

$$(a_i, a_j) \in \begin{cases} ECP^A & \text{if } |i-j| = 1 \\ DC^A & \text{if } |i-j| > 1 \end{cases} \qquad (1)$$

and thus by $\mathcal{A} \models \tau_n$ (see proposition 6) that $a_1, \ldots, a_n$ are the only components of a, i.e. $(z, a) \in COM^A$ is satisfied exactly by each $z \in \{a_1, \ldots, a_n\}$. Then, $\mathcal{A} \models \nu_2[a]$ immediately follows. Clearly, $a \in POL^A$, since by $0^A \notin P^A$ $a \neq 0^A$ and $a \leq^A \Phi_{1,1,2n,2n} \in P^A$. By (1) and $\mathcal{A} \models \sigma_n$ (see proposition 5) we have $\mathcal{A} \models \nu_1[a]$. Again by (1) both $\mathcal{A} \models \nu_4[a_1, a]$ and $\mathcal{A} \models \nu_5[a]$ hold.

To prove $\mathcal{A} \models \nu_3[a]$ suppose $v_1, v_2 \in A\backslash\{0^A\}$ satisfying $v_1 \cdot^A v_2 = 0^A$, $v_1 +^A v_2 = a$ and for any a_i either $a_i \leq^A v_1$ or $a_i \leq^A v_2$. Assuming $a_1 \leq^A v_1$ we define

$$q := \max\{i \in \mathbb{N} \mid a_1 \leq^A v_1, \ldots, a_i \leq^A v_1\}.$$

In case of $n = 1$ $\mathcal{A} \models \nu_3[a]$ holds trivially. Otherwise, from the conditions given above it follows $1 \leq q < n$ and thus, $a_{q+1} \leq^A v_2$. Hence by $(a_q, a_{q+1}) \in ECP^A$ we have indeed $\mathcal{A} \models \nu_3[a]$.

2) By similar arguments we can show $(b_1, b) \in SEQ^A$ and $x, y \in SDC^A$. Additionally, we have

$$(b_i, b_j) \in \begin{cases} ECP^A & \text{if } |i-j| = 1 \\ DC^A & \text{if } |i-j| > 1, \end{cases} \qquad (2)$$

and

$$(x_i, x_j), (y_i, y_j) \in DC^A \text{ if } i \neq j. \qquad (3)$$

The $b_1, \ldots, b_k$ are the only components of b, $x_1, \ldots, x_n$ the only components of x and $y_1, \ldots, y_n$ the only components of y.

3) Clearly, $x \leq^A a$, $y \leq^A a$, since $x_i, y_i \leq^A a_i$, and $a_1 \leq^A b_1$. To see that $x \leq^A b$, and analogously $y \leq^A b$, let $j \in \{1, \ldots, n\}$ and $p := \min\{q \mid |\alpha_{i_1}\cdots\alpha_{i_q}| \geq j\}$. Then, by definition of $b_p = \Phi_{e,f,g,h}$, we obtain $d_{e+1,1} +^A \cdots +^A d_{g,1} \leq^A b_p$ and $e+1 \leq 2j \leq g$, so that $x_j = d_{2j,1} \leq^A b_p$.

4) We have $\mathcal{A} \models \lambda_1[a, x, y]$ since $x_j \cdot^A a_i = 0^A$ and $y_j \cdot^A a_i = 0^A$ respectively if $j \neq i$, hence

$$a_i \cdot^A x = a_i \cdot^A x_i = x_i \text{ and } a_i \cdot^A y = a_i \cdot^A y_i = y_i. \quad (4)$$

5) Next we show $\mathcal{A} \models \lambda_2[a, b, x, y]$. This holds trivially if $k = 1$, since $(z_1, z_1) \in ECP^A$ implies $z_1 \cdot^A z_1 \neq 0^A$. Otherwise, we consider components b_j, b_{j+1} of b such that $(b_j, b_{j+1}) \in ECP^A$. Let $i := |\alpha_{i_1}\cdots\alpha_{i_j}|$. Then, $1 \leq i < n$ and, by (1), $(a_i, a_{i+1}) \in ECP^A$. Using (4) we obtain immediately from the definition of b_j and b_{j+1} that $a_i \cdot^A x = x_i \leq^A b_j$ and $a_{i+1} \cdot^A x = x_{i+1} \leq^A b_{j+1}$. Equally, $a_l \cdot^A y \leq^A b_j$ and $a_{l+1} \cdot^A y \leq^A b_{j+1}$ hold with $l := |\beta_{i_1}\cdots\beta_{i_j}|$, which verifies $\mathcal{A} \models \lambda_2[a, b, x, y]$.

6) Finally, we prove $\mathcal{A} \models \lambda_3[x, y, a, a_1, b, c]$. Let b_j be a component of b. We only show that $\mathcal{A} \models \Delta_{\alpha_{i_j}}[x, c, a, a_1, b_j]$ since the proof of $\mathcal{A} \models \Delta_{\beta_{i_j}}[y, c, a, a_1, b_j]$ is quite similar. For this, we define $q := |\alpha_{i_1}\cdots\alpha_{i_{j-1}}| + 1$ and $r := |\alpha_{i_1}\cdots\alpha_{i_j}|$, such that $a_q, \ldots, a_r$ are pairwise distinct components of a satisfying $(a_q, a_{q+1}), \ldots, (a_{r-1}, a_r) \in ECP^A$.

Suppose that $(a_1, s) \in SEQ^A$, $s \leq^A a$, $a_r \leq^A s$ for some $s \in A$ such that $a_i \leq^A s$ or $a_i \cdot^A s = 0^A$ for each a_i. Moreover, let $a_{l_1}, \ldots, a_{l_t}$ denote all components of a satisfying $a_i \leq^A s$. Then we get by $s = s \cdot^A a$

$$\begin{aligned} s &= s \cdot^A (a_1 +^A \cdots +^A a_n) \\ &= s \cdot^A (a_{l_1} +^A \cdots +^A a_{l_t}) \\ &= a_{l_1} +^A \cdots +^A a_{l_t}. \end{aligned}$$

By (1) and $\mathcal{A} \models \tau_t$ $a_{l_1}, \ldots, a_{l_t}$ are exactly the components of s. $(a_1, s) \in SEQ^A$ implies especially $a_1 \in \{a_{l_1}, \ldots, a_{l_t}\}$ and $\mathcal{A} \models \nu_3[s]$ (see proposition 8).

Next we show $\{a_1, \ldots, a_t\} = \{a_{l_1}, \ldots, a_{l_t}\}$, that is $s = a_1 +^A \cdots +^A a_t$. From this we get by $a_r \leq^A s$ immediately $q \leq r \leq t$, thus $a_q \leq^A s$. Let $p := \max\{i \mid a_1, \ldots, a_i \in \{a_{l_1}, \ldots, a_{l_t}\}\}$. If $p = t$ we are finished. Otherwise, if $1 \leq p < t$, we define

$$v_1 := a_1 +^A \cdots +^A a_p, \; v_2 := \sum_{i \in \{l_1, \ldots, l_t\}\backslash\{1, \ldots, p\}} a_i$$

such that $v_1, v_2 \neq 0^A$, $v_1 \cdot^A v_2 = 0^A$ and $v_1 +^A v_2 = a_{l_1} +^A \cdots +^A a_{l_t}$. Additionally, for any component a_{l_i} of s either $a_{l_i} \leq^A v_1$ or $a_{l_i} \leq^A v_2$ holds. From $\mathcal{A} \models \nu_3[s]$ it follows that $(z_1, z_2) \in ECP^A$ for some $z_1 \in \{a_1, \ldots, a_p\}$ and $z_2 \in \{a_{l_1}, \ldots, a_{l_t}\}\backslash\{a_1, \ldots, a_p\}$. Hence by (1), $z_1 = a_p$ and $z_2 = a_{p+1}$. But this is a contradiction to the definition of p, therefore $p = t$.

Because of the way b_j is defined $x_q, \ldots, x_r$ are the only x_i with $x_i \cdot^A b_j \neq 0^A$, hence by (4)

$$b_j \cdot^A x = x_q +^A \cdots +^A x_r = (a_q +^A \cdots +^A a_r) \cdot^A x.$$

The last property that remains to be shown is

$$a_{q+h-1} \cdot^A c \begin{cases} \neq 0^A & \text{if } \alpha_{i_j}[h] = 1 \\ = 0^A & \text{if } \alpha_{i_j}[h] = 0 \end{cases}$$

for any $h \in \{1, \ldots, |\alpha_{i_j}|\}$. This follows directly from the definition of $c_{q+h-1} = a_{q+h-1} \cdot^{A} c$ and the fact that

$$\pi[q + h - 1] = \pi[|\alpha_{i_1} \cdots \alpha_{i_{j-1}}| + h] = \alpha_{i_j}[h].$$

□

Using lemma 1 and 2 we can easily establish our undecidability result.

Theorem 4 *Any $T \subsetneq L_0^{S_\mathcal{M}}$ with $T_\mathcal{M} \subseteq T \subseteq Th(\mathcal{M})$ is undecidable. In particular, $Th(\mathcal{M})$ is undecidable.*

Proof: Let $T \subsetneq L_0^{S_\mathcal{M}}$ with $T_\mathcal{M} \subseteq T \subseteq Th(\mathcal{M})$ and C denote an arbitrary PCP-instance. We associate effectively to C the $S_\mathcal{M}$-sentence Ψ_C given in definition 7. Suppose $T \models \Psi_C$. Since $T \subseteq Th(\mathcal{M})$ is equivalent to $\mathcal{M} \models T$ we immediately get $\mathcal{M} \models \Psi_C$. Then, by lemma 1, C has a solution. Conversely, if C has a solution lemma 2 states that $T_\mathcal{M} \models \Psi_C$, from which $T \models \Psi_C$ follows by $T_\mathcal{M} \subseteq T$. Thus, we have

$$T \models \Psi_C \text{ iff } C \text{ has a solution,}$$

which implies the undecidablility of T. □

By theorem 3 we immediately get

Corollar *$Th(\mathcal{M})$ is neither elementarily axiomatizable nor recursively enumerable.* □

4 Conclusions

We have presented a model $\mathcal{M}$ of polygonal regions of the Euclidean plane. $\mathcal{M}$ turned out to be mereotopological, i.e. the topological relations usually provided by mereotopologies proved to be elementarily definable in $\mathcal{M}$. Thus, $Th(\mathcal{M})$ includes all elementary propositions on polygonal regions and their topological relations. Furthermore, we established the undecidability of $Th(\mathcal{M})$ by reduction from Post's correspondence problem. For this, we identified a small axiomatic fragment $T_\mathcal{M}$ of $Th(\mathcal{M})$, such that any T with $T_\mathcal{M} \subseteq T \subseteq Th(\mathcal{M})$ is undecidable. As a consequence, $Th(\mathcal{M})$ cannot be axiomatized in the first-order logic.

Let us finally discuss to what extent this result can be applied to differently defined mereotopological models. Obviously, the proof of theorem 4 works as well for three-dimensional regions which are defined similarly to the planar regions of $\mathcal{M}$: replace the squares shown in figure 4 with three-dimensional cubes arranged on the same plane. Probably, the closest mereotopological model to $\mathcal{M}$ is proposed by Pratt, Lemon and Schoop [14, 15]. Its complete axiomatization is grounded on an infinitary rule of inference allowing proofs of infinte length and is therefore without any consequence for its decision problem (in contrast to the elementary case). Although defined in a different way, its domain also consists of polygonal regions. It differs from $\mathcal{M}$ in the choice of taking "x is connected" as the primitive topological relation. Since in both models polygons are always finite sums of simple polygons we have reason to conjecture that our proof method applies as well.

To encode Post's correspondence problem in the way figure 4 shows it is crucial that finite sums of regions are definable. So, what about mereotopological models including sums of infinitely many regions ? In this case, finite sums can no longer be characterized by their boundedness. A method of defining finite sums can be obtained from Grzegorczyk's undecidability proof of closure algebra, i.e. Boolean algebra extended by some axioms for the predicate "x is closed" [10]. The main idea is to interpret the arithmetic of natural numbers, well-known as undecidable, by elementarily definable sets of regions and relations between regions. In particular, the natural number n is represented by the class of all regions of the underlying space that consist of exactly n components (then, equality between numbers corresponds to equivalence between regions with respect to the number of their components). To represent the set of all natural numbers in closure algebra Grzegorczyk defines a predicate which is satisfied only by regions composed of finitely many components. Consider the simple case of two disjoint regions x and y each of which consists entirely of possibly infinitely many components. Then, we can establish a one-to-one correspondence between their components iff there exists a region such that each of its components contains exactly one component of x and one component of y. x and y represent a natural number only if they consist of finitely many components. In fact, this is the case iff the set of components of x, minus one of its components, cannot be in a one-to-one correspondence to the set of components of y. Thus, finite sums of regions prove to be elementarily definable.

This way of defining finite sums of regions seems to apply to most mereotopological models. We therefore conjecture that Grzegorczyk's method or, alternatively, the method we use in theorem 4 for proving undecidability works as well for these models. Thus, it could be argued that the problem of first-order axiomatizations of spatial regions in terms of topological relations yields in general a negative solution. Moreover, Grzegorczyk's concept of encoding the arithmetic of natural numbers can also be used to show the undecidability of $Th(\mathcal{M})$ when modified in several ways. In comparison to this, our method has the advantage of being based upon few definability results for intuitive topological relations and few properties of polygonal regions.

Acknowledgements

This research was supported by the *Deutsche Forschungsgemeinschaft* (DFG) under grant Fr 806/7-1. I am indebted to Christian Freksa for his support and helpful comments on this work and to Carola Eschenbach for several valuable suggestions improving this paper. Also I would like to thank Bettina Berendt, Christie Manning and Reinhard Moratz for helpful comments on the final and an early version of this paper. The critiques and suggestions of two anonymous referees are gratefully acknowledged.

References

[1] N. Asher, L. Vieu. Toward a geometry of common sense: A semantics and complete axiomatization of mereotopology. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, Montreal, 846-852, 1995

[2] B. Bennett. Spatial reasoning with propositional logics. In J. Doyle, E. Sandewall, P. Torasso (eds.), *Principles of Knowledge Representation and Reasoning: Proceedings of the 4th International Conference (KR94)*, Morgan Kaufmann, San Francisco, CA, 51-62, 1994

[3] B. Bennett. Modal logics for qualitative spatial reasoning. *Bulletin of the Interest Group in Pure and Applied Logic (IGPL)*, 1995

[4] S. Borgo, N. Guarino, C. Masolo. A pointless theory of space based on strong connection and congruence. In L.C. Aiello, J. Doyle, S. Shapiro (eds.), *Principles of Knowledge Representation and Reasoning: Proceedings of the 5th International Conference (KR96)*, Morgan Kaufmann, Cambridge Massachusetts, 220-229, 1996

[5] K. Borsuk, W. Szmielew. *Foundations of Geometry*, North-Holland, 1960

[6] A.G. Cohn. Qualitative spatial representation and reasoning techniques. In G. Brewka, C. Habel, B. Nebel (eds.), *KI-97: Advances in Artificial Intelligence*, LNCS 1303, Springer, 1-30, 1997

[7] C. Dornheim. Unvollständigkeit topologischer Kalküle zum qualitativen räumlichen Schließen. Diplomarbeit, Fachbereich Informatik, Universität Hamburg, 1998

[8] H.-D. Ebbinghaus, J. Flum, W. Thomas. *Mathematical Logic*, Springer, 1984

[9] C. Eschenbach, W. Heydrich. Classical mereology and restricted domains. *International Journal of Human-Computer Studies*, 43, 723-740, 1995

[10] A. Grzegorczyk. Undecidability of some topological theories. *Fundamenta Mathematicae*, 38, 137-152, 1951

[11] J.E. Hopcroft, J.D. Ullman. *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, 1979

[12] S. Koppelberg. General theory of Boolean algebras. J.D. Monk (ed.), *Handbook of Boolean Algebras*, vol. 1, North-Holland, Amsterdam, 1989

[13] E. Moise. *Geometric Topology in Dimensions 2 and 3*, Springer, 1977

[14] I. Pratt, O. Lemon. Ontologies for plane, polygonal mereotopology. *Technical Report Series UMCS-97-1-1*, Department of Computer Science, University of Manchester, 1997

[15] I. Pratt, D. Schoop. A complete axiom system for polygonal mereotopology of the real plane. *Technical Report Series UMCS-97-2-2*, Department of Computer Science, University of Manchester, 1997

[16] D.A. Randell, Z. Cui, A.G. Cohn. 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 3rd International Conference (KR92)*, Morgan Kaufmann, San Mateo, 165-176, 1992

[17] J. Renz, B. Nebel. On the complexity of qualitative spatial reasoning: a maximal tractable fragment of the region connection calculus. In *Proceedings of the 15th International Conference on Artificial Intelligence (IJCAI97)*, Nagoya, Japan, 1997

[18] A. Tarski. Foundations of the geometry of solids. *Logic, Semantics, Metamathematics*, Clarendon Press, Oxford, 24-30, 1956

[19] A.C. Varzi. Parts, wholes, and part-whole relations: The prospects of mereotopology. *Data and Knowledge Engineering*, 20, 259-286, 1996

[20] A.N. Whitehead. *Process and Reality. An Essay in Cosmology*, MacMillan, New York, 1929

Complexity of Reasoning

Complexity results for independence and definability in propositional logic

Jérôme Lang
IRIT-UPS
118 route de Narbonne, 31062 Toulouse Cedex
France
e-mail: lang@irit.fr

Pierre Marquis
CRIL/Université d'Artois
rue de l'Université, S.P. 16, 62307 Lens Cedex
France
e-mail: marquis@cril.univ-artois.fr

Abstract

This paper focuses on three notions of independence/dependence that relate formulas and variables inside propositional logic: conditional (variable-variable) independence, formula-variable independence and a form of dependence between sets of variables w.r.t. a formula called definability. For each notion, definitions, various formal characterizations, complexity results and some tractable restrictions are exhibited. Close connections to some notions of relevance recently introduced in the literature are pointed out as well.

1 Introduction

When reasoning about knowledge represented in propositional logic, it is often useful to identify some particular relationships between several parts of the knowledge base, or between some of the variables. The first kind of relationship one may have in mind is *independence*, which is particularly useful when reasoning about belief change [11] [9], about action [19] or causality. At the opposite side, one may be interested in identifying dependence relations between variables or formulas.

This paper aims at studying several forms of independence/dependence that are ***built-in inside propositional logic*** (this contrasts with many works where independence is extra-logical); we especially consider these forms of independence from the point of view of computational complexity.

After some formal preliminaries (Section 2), we start with a simple form of formula-variable independence (Section 3). A knowledge base Σ is viewed independent of a set X of variables whenever giving truth values to variables of X cannot change the contents of Σ. Then we focus on conditional independence (Section 4). Conditional independence relates sets of variables w.r.t. a knowledge base and possible contexts (represented as assignments of variables). Intuitively, two variables x and y are considered independent iff for every possible context ω_Z, knowing the truth value of x does not enable deriving the truth value of y given Σ and *vice-versa*. We first consider conditional independence as introduced in [5] [8], and we extend this notion to two stronger forms of independence by enabling incomplete knowledge about contexts to be handled. We show how these different notions of independence are interrelated. Some characterization results are provided, the complexity of independence checking is analyzed and some tractable restrictions are exhibited. The computational complexity of closely related forms of (in)dependence, especially influenceability [4], relevance [14] [15], and causal independence [8] is also investigated.

Then we turn to the opposite side and focus on a form of dependence, called definability (Section 5). Intuitively, given a knowledge base Σ, a variable y can be defined from a set X of variables iff the knowledge of the truth values of the variables of X (whatever they are) enables us to conclude about the truth value of y. Several characterizations of defining and minimal defining families are provided; we also show how definability interacts with independence. Additionally, the complexity of basis checking is studied in the general case and some restricted cases where it is polynomial are pointed out.

Finally, some applications of independence / dependence are briefly sketched (Section 6), before concluding (Section 7).

2 Formal Preliminaries

Let PS be a finite set of propositional variables. $PROP_{PS}$ is the propositional language built up from PS, the connectives and the Boolean constants $true$ and $false$ in the usual way. For every $X \subseteq PS$, $PROP_X$ denotes the sublanguage of $PROP_{PS}$ generated from the variables of X, only. A literal of $PROP_X$ is either a variable of X or the negation of a variable of X. A clause δ (resp. a term γ) of $PROP_X$ is a (possibly empty) disjunction (resp. conjunction) of literals of $PROP_X$. A CNF (resp. a DNF) formula of $PROP_X$ is a conjunction of clauses (resp. a disjunction of terms) of $PROP_X$.

From now on, Σ denotes a finite propositional knowledge base, i.e. a conjunctively interpreted finite set of propositional formulas from $PROP_{PS}$. $Var(\Sigma)$ is the set of propositional variables appearing in Σ. Elements of PS are denoted x, y etc. Subsets of PS are denoted X, Y etc. In order to simplify notations, we will assimilate every singleton $X = \{x\}$ with its unique element x. Full instantiations of variables of $X \subseteq PS$ are denoted by ω_X and their set is denoted Ω_X.

For every formula Φ and every variable x, $\Phi_{x\leftarrow 0}$ (resp. $\Phi_{x\leftarrow 1}$) is the formula obtained by replacing in Φ every occurence of x by the constant $false$ (resp. $true$). Two formulas Ψ and Φ are said *equivalent modulo* a formula Σ iff $\Sigma \wedge \Psi \equiv \Sigma \wedge \Phi$.

The set of prime implicants modulo Σ of a formula Φ over $PROP_X$, which will be denoted by $PI^X_\Sigma(\Phi)$, is defined by $PI^X_\Sigma(\Phi) = max(\{PI(\Sigma \Rightarrow \Phi) \cap PROP_X\}, \models)$ where $PI(\Phi)$ $(= PI^{PS}_{true}(\Phi))$ denotes the set of prime implicants of Φ.
Dually, $IP^X_\Sigma(\Phi)$ denotes the set of prime implicates of Φ modulo Σ over $PROP_X$, i.e.,
$IP^X_\Sigma(\Phi) = min(\{IP(\Sigma \Rightarrow \Phi) \cap PROP_X\}, \models)$
where $IP(\Phi)$ $(= IP^{PS}_{true}(\Phi))$ denotes the set of prime implicates of Φ. In both $PI^X_\Sigma(\Phi)$ and $IP^X_\Sigma(\Phi)$, only one representative per equivalence class is kept.

For instance, if $\Sigma = \{a \vee b, \neg a \wedge c \Rightarrow e, d \Leftrightarrow e\}$, then $PI^{\{a,b,c,d\}}_\Sigma(e) = \{d, \neg a \wedge c, \neg a \wedge \neg b\}$, $PI^{\{a,c\}}_\Sigma(e) = \{\neg a \wedge c\}$, $PI^{\{b,c\}}_\Sigma(e) = \emptyset$, $PI^{\{a,b,c,d\}}_\Sigma(a \vee b) = \{true\}$, $IP(\Sigma) = \{a \vee b, a \vee \neg c \vee e, \neg d \vee e, d \vee \neg e, a \vee \neg c \vee d\}$.

The complexity results we give in this paper refer to some complexity classes which deserve some recalls. More about them can be found in Papadimitriou's textbook [24]. We assume that the classes P, NP and coNP are known to the reader. The following classes will also be considered:
(i) 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: a pair of formulas $\langle \varphi, \psi \rangle$ is in SAT−UNSAT iff φ is satisfiable and ψ is not.
(ii) $\Sigma^P_2 = NP^{NP}$ is the class of all languages recognizable in polynomial time by a nondeterministic Turing machine using NP-oracles. Lastly, $\Pi^P_2 = co\Sigma^P_2$.

3 Formula-variable independence

In this section, we investigate several forms of formula-variable (in)dependence (FV (in)dependence for short), starting with the simplest one.

Let Σ be a KB and X a finite subset of PS. The most trivial form of formula-variable independence one can have in mind is the following syntactical one.

Definition 1 (syntactical FV independence)
Σ is syntactically independent of X iff $Var(\Sigma) \cap X = \emptyset$. When Σ is not syntactically independent of X, it is said syntactically dependent *of X.*

This basic form of independence suffers from two important drawbacks. On the one hand, it does not satisfy the principle of irrelevance of syntax: two equivalent formulas are not independent of the same variables. On the other hand, it is too much demanding: while it is the case that no variable Σ is independent of which has any influence on Σ, the converse does not hold in the general case. For instance, $\Sigma = \{a, b \Leftrightarrow b\}$ is syntactically dependent of b, but any information about b does not convey anything new about Σ. This calls for a weaker but more robust notion of FV independence.

Definition 2 ((semantical) FV independence)
Σ is independent *of X, noted $X \not\mapsto \Sigma$, iff there exists a formula Σ' s.t. $\Sigma' \equiv \Sigma$ and Σ is syntactically independent of X. When Σ is not independent of X, it is said* dependent *of X.*

Thus, Σ is independent of X iff Σ can be made syntactically independent of X while preserving logical equivalence. Accordingly, syntactical indepeadence implies independence, but the converse does not hold in the general case. Thus, $\Sigma = \{a, b \Leftrightarrow b\}$ is independent of b.

Clearly enough, FV independence exhibits the following metatheoretic properties:

Proposition 1
(1) If $\Sigma \equiv \Sigma'$, then ($X \not\mapsto \Sigma$ iff $X \not\mapsto \Sigma'$).
(2) If $X \not\mapsto \Sigma$ and $X' \subseteq X$, then $X' \not\mapsto \Sigma$.

(3) If $X \not\mapsto \Sigma$ and $X \not\mapsto \Sigma'$, then $X \not\mapsto \neg\Sigma$, $X \not\mapsto \Sigma \wedge \Sigma'$ and $X \not\mapsto \Sigma \vee \Sigma'$.

Nevertheless, FV independence neither is monotonic nor anti-monotonic w.r.t. expansion of Σ (strengthening or weakening Σ can easily make it no longer independent of X). It is also not monotonic w.r.t. expansion of X. Thus, while $\Sigma = \{a\}$ is independent of b, neither $\{a, b\}$ nor $\{a \vee b\}$ are independent of b. Moreover, Σ is not independent of $\{a, b\}$.

Finally:

Proposition 2 $X \not\mapsto \Sigma$ *iff* $\forall x \in X$, $x \not\mapsto \Sigma$.

Dually, when Σ is dependent of X, there exists an $x \in X$ Σ is dependent of which. The case where Σ depends on every variable of X calls for a stronger notion of FV dependence, called full FV dependence.

Definition 3 (full FV dependence) Σ *is* fully dependent of X *iff Σ is dependent of every variable $x \in X$.*

A particular case of full FV dependence that proves useful is reached when $X = Var(\Sigma)$. In this situation, Σ is said simplified.

Definition 4 (simplified KB) Σ *is* simplified *iff Σ is fully dependent of $Var(\Sigma)$.*

For instance, $\Sigma = \{a, b \Leftrightarrow b\}$ is not simplified while the equivalent KB $\Sigma' = \{a\}$ is simplified.

Simplified KBs do not incorporate any useless variable. Accordingly, simplifying a KB can prove helpful for various reasoning tasks since it enables improving its readibility while decreasing its size. As the next proposition shows it, the notion of simplified KB actually is the point where syntactical independence and (semantical) independence coincide.

Proposition 3 *Σ is simplified iff the following equivalence holds: for every $X \subseteq PS$, Σ is syntactically independent of X iff $X \not\mapsto \Sigma$ holds.*

Interestingly, every KB can be simplified in a truth-preserving way:

Proposition 4 *For every Σ, there exists a simplified KB Σ' s.t. $\Sigma \equiv \Sigma'$.*

The following result gives a constructive proof of the previous proposition. It tells us that Σ remains unchanged when a variable Σ is independent of which is assigned a truth value:

Proposition 5 *The next four statements are equivalent:*
(1) $x \not\mapsto \Sigma$;
(2) $\Sigma_{x \leftarrow 0} \equiv \Sigma_{x \leftarrow 1}$;
(3) $\Sigma \equiv \Sigma_{x \leftarrow 0}$;
(4) $\Sigma \equiv \Sigma_{x \leftarrow 1}$.

On the previous example, we have $\Sigma_{b \leftarrow 0} \equiv \Sigma_{b \leftarrow 1} \equiv a$. Accordingly, Σ is independent of b.

Thanks to the proposition above, we can easily design a greedy algorithm for simplifying KBs. This algorithm consists in considering every variable x of $Var(\Sigma)$ in a successive way, while replacing Σ by $\Sigma_{x \leftarrow 0}$ whenever Σ is independent of x. At the end, the resulting KB is simplified.

We now give several characterizations of FV independence, allowing us to relate it with some notions of independence pointed out so far. 2. and 3. of Proposition 6 below tell us that FV independence can be easily read off from Σ whenever Σ is given under the prime implicants or the prime implicates normal form. For KBs given under these normal forms, independence comes down to syntactical independence.

Proposition 6 *The next four statements are equivalent:*
(1) $X \not\mapsto \Sigma$;
(2) $PI(\Sigma) \subseteq PROP_{Var(\Sigma) \setminus X}$;
(3) $IP(\Sigma) \subseteq PROP_{Var(\Sigma) \setminus X}$;
(4) $\Sigma \equiv forget(\Sigma, X)$.

Stepping back to our running example, $PI(\Sigma) = IP(\Sigma) = \{a\}$ clearly is syntactically independent of variable b.

Interestingly, this proposition shows our notion of FV independence to coincide with Boutilier's notion of non-influenceability; indeed, Proposition 4 from [4] states that Σ is influenceable iff there exists a prime implicant of Σ which contains a variable from X, where X is the set of so-called controllable variables. It also shows FV independence to coincide with Lakemeyer's notion of irrelevance of a formula to a subject matter since Theorem 3 from [14] states that Σ is relevant to X iff there exists a prime implicate of Σ which contains a variable from X. Accordingly, Proposition 6 gives rise to a new model-theoretic characterization of influenceability and relevance. The last characterization result 4. of Proposition 6 relates FV independence to the *forget* operator introduced in the more general framework of first-order logic by Lin and Reiter [17]. $forget(\Sigma, X)$ is defined inductively by:
$forget(\Sigma, \emptyset) = \Sigma$;

$forget(\Sigma, \{x\}) = (\Sigma_{x \leftarrow 0} \vee \Sigma_{x \leftarrow 1})$;
$forget(\Sigma, X \cup \{x\}) = forget(forget(\Sigma, X), \{x\})$.

$forget(\Sigma, X)$ is the most general consequence of Σ that is independent of X, up to logical equivalence. On our example, $forget(\{a, b \Leftrightarrow b\}, \{b\}) \equiv a$.

Point 3. of Proposition 6 also allows us to give an alternative characterization of the notion of *strict relevance* introduced by Lakemeyer in [15]. Indeed, a formula Σ is said strictly relevant[1] to a subject matter X iff there exists a prime implicate of Σ mentioning a variable from X, and every prime implicate of Σ mentions only atoms from X. Accordingly, 3. shows Σ strictly relevant to X iff Σ is dependent of X and independent of $Var(\Sigma) \setminus X$.

We turn now into computational complexity issues. While syntactical FV dependence can be easily checked in linear time in the size of the input, this is far from being expected for FV dependence and full FV dependence in the general case:

Proposition 7 (complexity of FV dependence) FV DEPENDENCE *and* FULL FV DEPENDENCE *are* NP-*complete.*

Using then Proposition 6, we get the following corollary:

Proposition 8
(1) Boutilier's influenceability is NP-*complete.*
(2) Lakemeyer's relevance of a formula to a subject matter is NP-*complete.*

We have also derived the following:

Proposition 9
(1) Lakemeyer's [15] strict relevance of a formula to a subject matter is BH_2-*complete.*
(2) Lakemeyer's [14] strict relevance of a formula to a subject matter is Π_2^p-*complete.*

These complexity results improve Theorem 50 from [15] which only points out the NP-hardness of irrelevance and strict irrelevance (w.r.t. the definition given in [15]).

Interestingly, the complexity of both decision problems fall down to P whenever checking FV (in)dependence becomes tractable. Apart from the case of syntactical FV independence, some other restrictions on Σ makes FV (in)dependence testable in polynomial time. Especially, as a corollary of Proposition 5 we get:

Proposition 10 *Whenever Σ belongs to a class $\mathcal{C}$ of formulas that is tractable for clausal query answering and stable for variable instantiation (i.e. replacing in $\Sigma \in \mathcal{C}$ any variable by* true *or by* false *gives a formula that still belongs to $\mathcal{C}$) then* FV (IN)DEPENDENCE *is in* P.

In particular, when Σ is restricted to a renamable Horn CNF formula or to binary clauses, FV (IN)DEPENDENCE belongs to P. Theorem 51 from [15] can be viewed as a restriction of the latter to the case where Σ is Horn.

Additionally, when FV INDEPENDENCE is in P, the simplification algorithm given above runs in time polynomial in the size of Σ, i.e. the corresponding function problem is in FP: the simplification task can be achieved in a tractable way.

4 Conditional independence

In contrast to FV independence, conditional independence relates several sets of variables within a propositional KB. Conditional independence has been first introduced by Darwiche and Pearl [8] [6] as a logical counterpart of probabilistic independence.

Definition 5 (conditional independence) *[8] Let Σ be a KB and X, Y, Z disjoint subsets of PS. X and Y are independent given Z w.r.t. Σ (denoted by $X \sim_\Sigma^Z Y$) iff $\forall \omega_X \in \Omega_X, \forall \omega_Y \in \Omega_Y, \forall \omega_Z \in \Omega_Z$, the consistency of both $\omega_X \wedge \omega_Z \wedge \Sigma$ and $\omega_Y \wedge \omega_Z \wedge \Sigma$ implies the consistency of $\omega_X \wedge \omega_Y \wedge \omega_Z \wedge \Sigma$.*

Clearly enough, conditional independence given Z w.r.t. Σ satisfies the following properties [7]:

Proposition 11
(1) $X \sim_\Sigma^Z Y$ iff $Y \sim_\Sigma^Z X$.
(2) If $\Sigma \equiv \Sigma'$, then ($X \sim_\Sigma^Z Y$ iff $X \sim_{\Sigma'}^Z Y$).
(3) If $X' \subseteq X$ and $Y' \subseteq Y$ and $X \sim_\Sigma^Z Y$, then $X' \sim_\Sigma^Z Y'$.

However, conditional independence neither is not stable by contraction or by expansion of Z. For instance, let $\Sigma = \{\neg a \vee \neg b \vee c, \neg a \vee b \vee d, a \vee \neg c, \neg a \vee c \vee y, b \vee \neg c \vee d\}$. We have $c \sim_\Sigma^\emptyset d$ but $c \not\sim_\Sigma^{\{a\}} d$ (because of $\neg a \vee c \vee d$: when a is *true*, learning $\neg c$ tells that d is *true*), $c \not\sim_\Sigma^{\{b\}} d$ and however $c \sim_\Sigma^{\{a,b\}} d$ (because $a \wedge b \wedge \Sigma \models c$, $a \wedge \neg b \wedge \Sigma \models d$ and $\neg a \wedge \Sigma \models \neg c$, i.e., full knowledge about a and b breaks all links between c and d). Conditional independence is also not stable by weakening

[1]As pointed out in [15], this notion differs from the one given in [14], where Σ is said strictly relevant to X iff every prime implicate of Σ contains a variable from X.

or strengthening Σ in the general case. Thus, while we have $c \sim_{\Sigma}^{\emptyset} d$, we also have $c \not\sim_{\Sigma\cup\{c\Leftrightarrow d\}}^{\emptyset} d$; while we have $c \sim_{\Sigma}^{\{a,b\}} d$, we also have $c \not\sim_{\Sigma\setminus\{\neg a\vee\neg b\vee c\}}^{\{a,b\}} d$.

The two limit cases when Z is respectively empty or equal to $Var(\Sigma) \setminus (X \cup Y)$, are of particular interest, especially when investigating computational complexity.

Definition 6 (marginal independence) *X and Y are* marginally independent *w.r.t. Σ iff $X \sim_{\Sigma}^{\emptyset} Y$.*

Definition 7 (ceteris paribus independence) *X and Y are* ceteris paribus independent *w.r.t. Σ (denoted by $X \sim_{\Sigma}^{ceteris\ paribus} Y$) iff $X \sim_{\Sigma}^{Var(\Sigma)\setminus(X\cup Y)} Y$.*

As said in [8], $X \sim_{\Sigma}^{Z} Y$ holds iff for any possible *full information* about Z, the addition of information about Y does not tell us anything new about X. This definition does not apply to contexts where the new information that can be learned about Z may be incomplete., i.e. the truth value of some variables of Z is not available, or, more generally, many partial (and eventually mutually exclusive) assignments of variables of Z are possible. For instance, if Z represents a set of possibly measurable variables, associated to a set of sensors (one for each $z \in Z$), it can be the case that some measurements fail, i.e. the value of z is not always available.

We are now going to strengthen Darwiche and Pearl's conditional independence by taking account of incomplete information about Z. We propose two definitions, whether the incomplete information about Z is required to be conjunctive or not. Namely, X and Y are strongly (resp. perfectly) independent given Z w.r.t. Σ iff *whichever conjunctive information (i.e. a set of facts)* (resp. *whichever information*, i.e. any formula) we may learn about Z, then the addition of information about Y does not enable telling anything new about X.

Definition 8 (strong conditional independence) *X and Y are* strongly independent *given Z w.r.t. Σ (denoted $X \approx_{\Sigma}^{Z} Y$) iff for any term γ_Z of $PROP_Z$, the consistency of both $\omega_X \wedge \gamma_Z \wedge \Sigma$ and of $\omega_Y \wedge \gamma_Z \wedge \Sigma$ implies the consistency of $\omega_X \wedge \omega_Y \wedge \gamma_Z \wedge \Sigma$.*

Strong conditional independence has the same metatheoretic properties as conditional independence, plus the preservation by contraction of Z (which is a trivial consequence of the definition).

Marginal strong conditional independence obviously coincides with marginal conditional independence. *Ceteris paribus* strong conditional independence is defined by imposing $Z = Var(\Sigma) \setminus (X \cup Y)$ and denoted by $X \approx_{\Sigma}^{ceteris\ paribus} Y$.

Since the set of all possible choices for γ_Z corresponds to the set of all partial assignments of the variables of Z, we get:

Proposition 12
$X \approx_{\Sigma}^{Z} Y$ iff $X \sim_{\Sigma}^{Z'} Y$ for every $Z' \subseteq Z$.

Obviously, $X \approx_{\Sigma}^{Z} Y$ entails $X \sim_{\Sigma}^{Z} Y$. The converse generally does not hold since conditional independence is not stable by contraction of Z. Indeed, for the previous example, we have $c \sim_{\Sigma}^{\{a,b\}} d$ but $c \not\approx_{\Sigma}^{\{a,b\}} d$ since $c \sim_{\Sigma}^{\{b\}} d$ does not hold.

As a corollary of Proposition 12, *ceteris paribus* strong independence coincides with Lakemeyer's definition [14] of irrelevance between two subject matters relative to a knowledge base[2].

The following results enable characterizing strongly conditionally independent sets of variables. They both express that $X \approx_{\Sigma}^{Z} Y$ iff any set of simple facts (i.e. literals) we may learn about Z never enables deducing a nontrivial *disjunctive information* involving both X and Y.

Proposition 13 (consequence decomposability) *$X \approx_{\Sigma}^{Z} Y$ iff for any term γ_Z of $PROP_Z$, and $\forall\varphi_X \in PROP_X$, $\forall\varphi_Y \in PROP_Y$, $\gamma_Z \wedge \Sigma \models \varphi_X \vee \varphi_Y$ implies $\gamma_Z \wedge \Sigma \models \varphi_X$ or $\gamma_Z \wedge \Sigma \models \varphi_Y$.*

Proposition 14 *$X \approx_{\Sigma}^{Z} Y$ iff $\forall\delta \in IP_{true}^{X\cup Y\cup Z}(\Sigma)$, δ never includes both a variable of X and a variable of Y.*

Proposition 14 slightly generalizes Lakemeyer's Theorem 5 from [14] (we recover it when $Z = V \setminus (X \cup Y)$). Note that there is no such nice characterization in terms of prime implicates for (standard) conditional independence.

As a consequence of Proposition 14, it is enough in practice to consider strong conditional independence of single variables:

Proposition 15
$X \approx_{\Sigma}^{Z} Y$ iff $\forall x \in X\ \forall y \in Y$, $x \approx_{\Sigma}^{Z\cup(X\setminus\{x\})\cup(Y\setminus\{y\})} y$.

This result is useful for the practical computation of strong conditional independence relations. Note that

[2]We omit Lakemeyer's original definition of relevance for the sake of brevity; it is enough to refer to Theorem 31 in [15] which states that X is relevant to Y w.r.t. Σ iff there is a Z such as $X \not\sim_{\Sigma}^{Z} Y$.

there is no similar result for (standard) conditional independence. Things become even simpler with *ceteris paribus* strong independence, since Proposition 15 becomes: $X \approx_{\Sigma}^{ceteris\ paribus} Y$ iff $\forall x \in X$ $\forall y \in Y$, $x \approx_{\Sigma}^{ceteris\ paribus} y$ (cf. Lemma 15 in [15]).

Interestingly, FV independence can be recovered from strong conditional independence:

Proposition 16
Let new be a variable of $PS \setminus Var(\Sigma)$.
$X \not\hookrightarrow \Sigma$ *iff* $X \approx_{\Sigma \Leftrightarrow new}^{Var(\Sigma)\setminus X} new$.

This means that in any state of knowledge regarding $Var(\Sigma) \setminus X$, knowing the truth values of variables in X cannot help us knowing the truth value of *new* and hence of Σ. The converse, i.e., expressing strong conditional independence from FV independence, is possible as well (see Proposition 17). However, the exhibited transformation is not polynomial and thus will not be helpful when investigating computational complexity.

According to Proposition 14, $x \not\approx_{\Sigma}^{Z} y$ iff there is a prime implicate δ in $IP_{true}^{Z\cup\{x,y\}}(\Sigma)$ mentioning both x and y. This is equivalent to say that there is a prime *implicant* γ in $PI_{\Sigma}^{Z\cup\{x\}}(y)$ or in $PI_{\Sigma}^{Z\cup\{x\}}(\neg y)$, *consistent with* Σ and mentioning x. The consistency condition is necessary; indeed, let us consider $\Sigma = \{c \Rightarrow a, d \Rightarrow b\}$ and $Z = \{c, d\}$; $PI_{\Sigma}^{Z\cup\{a\}}(b) = \{c \wedge \neg a, d\}$ mentions a but nevertheless $a \approx_{\Sigma}^{\{c,d\}} b$ holds; this is because $c \wedge \neg a$ is not consistent with Σ, or in other words, $c \wedge \neg a$ is a prime implicant of $\Sigma \Rightarrow b$ only because it is a prime implicant of $\neg\Sigma$. Thus, the set of prime implicants of interest is $PI_{\Sigma}^{Z\cup\{x\}}(y) \setminus PI^{Z\cup\{x\}}(\neg\Sigma)$, which corresponds exactly to set of minimal abductive explanations of y w.r.t. Σ, where the set of possible individual hypotheses is the set of literals built up from $Z \cup \{x\}$ [10]. Equivalently, this set is the *label* of y according to the ATMS literature [25]. This leads to the following characterization:

Proposition 17 *Let* $Label_{\Sigma}^{Z\cup\{x\}}(y)$ *be the disjunction of all prime implicants* γ *in* $PI_{\Sigma}^{Z\cup\{x\}}(y)$ *s.t.* $\gamma \wedge \Sigma$ *is consistent. Then* $x \approx_{\Sigma}^{Z} y$ *iff both* $Label_{\Sigma}^{Z\cup\{x\}}(y)$ *and* $Label_{\Sigma}^{Z\cup\{x\}}(\neg y)$ *are independent of* x.

In other words, $x \approx_{\Sigma}^{Z} y$ iff both x and $\neg x$ are irrelevant hypotheses for (minimally) explaining y and $\neg y$ (i.e. neither x nor $\neg x$ does participate to any minimal explanation of y and neither x nor $\neg x$ does participate to any minimal explanation of $\neg y$) [10].

This gives us an algorithm for computing strong independence relations using a basic ATMS. Let $SIV_{\Sigma}^{Z}(y) = \{x \in Var(\Sigma) \setminus \{y\} \mid x \approx_{\Sigma}^{Z} y\}$. A list of variables L is initialized to $Var(\Sigma) \setminus \{y\}$, and each time a new environment of y (i.e. one of the disjuncts of $Label_{\Sigma}^{Z\cup\{x\}}(y)$) or of $\neg y$ is computed, then all variables appearing in it are removed from L. Then, at any step, L contains $SIV_{\Sigma}^{Z}(y)$ and the algorithm reaches $SIV_{\Sigma}^{Z}(y)$ when it ends up (this shows a possible "anytime" use of this algorithm).

We now define our last notion of conditional independence, stronger than the two previous ones.

Definition 9 (perfect conditional independence) *X and Y are perfectly independent for Z w.r.t.* Σ *(denoted* $X \asymp_{\Sigma}^{Z} Y$*) iff* $\forall \varphi_Z \in PROP_Z$, $\forall \omega_X \in \Omega_X$, $\forall \omega_Y \in \Omega_Y$, *the consistency of both* $\omega_X \wedge \Sigma \wedge \varphi_Z$ *and of* $\omega_Y \wedge \Sigma \wedge \varphi_Z$ *implies the consistency of* $\omega_X \wedge \omega_Y \wedge \Sigma \wedge \varphi_Z$.

$X \asymp_{\Sigma}^{Z} Y$ means that no significant relationship between X and Y can be inferred when learning *any* information, including disjunctive information, about Z. This is expressed by the following result, similar to Proposition 13:

Proposition 18
$X \asymp_{\Sigma}^{Z} Y$ *iff* $\forall \varphi_Z \in PROP_Z$, $\forall \varphi_X \in PROP_X$, $\forall \varphi_Y \in PROP_Y$, $\varphi_Z \wedge \Sigma \models \varphi_X \vee \varphi_Y$ *implies* $\varphi_Z \wedge \Sigma \models \varphi_X$ *or* $\varphi_Z \wedge \Sigma \models \varphi_Y$.

Clearly, $X \asymp_{\Sigma}^{Z} Y$ implies $X \approx_{\Sigma}^{Z} Y$ (the latter is recovered when φ_Z is a term). The converse is generally false, as shown by the following example: $\Sigma = \{c \Rightarrow a, d \Rightarrow b\}$, $X = \{a\}$, $Y = \{b\}$, $Z = \{c, d\}$, $\varphi_X = a$, $\varphi_Y = b$, and $\varphi_Z = c \vee d$. We have $a \approx_{\Sigma}^{Z} b$; nevertheless, we have $\varphi_Z \wedge \Sigma \not\models a$, $\varphi_Z \wedge \Sigma \not\models b$ and $\varphi_Z \wedge \Sigma \models a \vee b$, which means that $a \not\asymp_{\Sigma}^{Z} b$.

In contrast to the two weaker forms of independence discussed before, perfect independence is insensitive to the granularity of the representation, in the following sense: if Z is replaced by another set of variables Z' s.t. Z and Z' define each other in Σ (see Section 5), then perfect independence is preserved:

Proposition 19 *If Z and Z' are subsets of PS s.t. every z of Z (resp. Z') is equivalent modulo* Σ *to a formula of* $PROP_{Z'}$ *(resp.* $PROP_Z$*), then (*$X \asymp_{\Sigma}^{Z} Y$ *iff* $X \asymp_{\Sigma}^{Z'} Y$*).*

As an illustration, let $Z = \{n(orth), s(outh), e(ast), w(est)\}$ and $Z' = \{ne, nw, se, sw\}$ where Σ

contains $s \Leftrightarrow (se \vee sw)$, $e \Leftrightarrow (ne \vee se)$, etc. and mutual exclusivity statements between ne,nw,se and sw (such as $sw \Rightarrow \neg se$, etc.). Z and Z' define each other, because Σ entails $ne \Leftrightarrow (n \wedge e)$, etc. Let us now add to Σ the two formulas $se \Rightarrow rain$ and $sw \Rightarrow wind$, which imply $s \Rightarrow (rain \vee wind)$. Then $rain$ and $wind$ are strongly independent given Z' while they are not given Z. In both cases, perfect independence between $rain$ and $wind$ does not hold, because we may later learn something about the theme "location" implying a link between $wind$ and $rain$. This example shows that the lack of perfect independence between X and Y corresponds intuitively to a *potential dependence* given the theme corresponding to Z. Unfortunately, it seems that perfect conditional independence does not have any simple characterization in terms of prime implicants/implicates.

We end up this section with computational complexity issues. We start by investigating in depth the complexity of standard conditional independence. We distinguish a number of restrictions on X, Y, Z and Σ which may lower the complexity level, namely: $|X| = 1$ and/or $|Y| = 1$ (checking whether a variable is independent from a variable / a set of variables), $X \cup Y = Var(\Sigma)$ (*twofold partition independence*), $Z = \emptyset$ (*marginal independence*) and $Z = Var(\Sigma) \setminus (X \cup Y)$ (*ceteris paribus independence*)[3].

Proposition 20 (complexity of independence) *The results are synthesized on Table 1 (where V stands for $Var(\Sigma)$).*

The numerous results contained in Proposition 20 are proved in the following order, which tries to minimize the number of proofs: 1. CONDITIONAL INDEPENDENCE is in Π_2^P; 2. MARGINAL INDEPENDENCE OF A VARIABLE FROM A SET OF VARIABLES is Π_2^P-hard; 3. CONDITIONAL INDEPENDENCE OF SINGLE VARIABLES is Π_2^P-hard; 4. CETERIS PARIBUS INDEPENDENCE is in coNP; 5. TWOFOLD PARTITION INDEPENDENCE is coNP-hard; 6. MARGINAL VARIABLE INDEPENDENCE is coBH$_2$-complete.

Proposition 21 (strong independence) STRONG CONDITIONAL INDEPENDENCE *is Π_2^P-complete.*

Strong conditional independence remains Π_2^P-complete when X or Y is a singleton and when both are singletons. As to *ceteris paribus* strong independence, it is in Π_2^P but we do do not have a proof of completeness.

[3]Note that, for twofold partition independence, the distinctions on Z are irrelevant; therefore, all three problems of the last row of Table 1 are identical.

Note that the Π_2^P-completeness of STRONG CONDITIONAL INDEPENDENCE coheres with the Σ_2^P-completeness of checking whether an individual hypothesis is relevant (for minimal explanation), as shown in [10]. More interestingly, the abductive characterization[4] (Proposition 15) of strong conditional independence enables taking advantage of some restrictions (especially restricting Σ to a set of Horn clauses) for which the computational complexity of checking irrelevance for minimal explanation falls down to the first level of the polynomial hierarchy, carrying with it the complexity of strong conditional independence. Considering DNF KBs is another restriction that makes the complexity of STRONG CONDITIONAL INDEPENDENCE falling down to the first level of the polynomial hierarchy. To be more precise, when Σ is in DNF, we can easily prove that STRONG CONDITIONAL INDEPENDENCE becomes coNP-complete.

Things are not very different with perfect conditional independence.

Proposition 22 (perfect independence) PERFECT CONDITIONAL INDEPENDENCE *is Π_2^P-complete.*

Focusing on the aforementioned syntactic restrictions leads exactly to the same results as for strong conditional independence, and the same lack of proof of Π_2^P-completeness for the *ceteris paribus* case.

These results are significant for a number of related problems such as relevance or causality (with applications to model-based diagnosis [6]). In particular, we identify the complexity of causal independence in symbolic causal networks (Darwiche and Pearl [8]). We omit the full definition of causal independence because it would need too much space (see [8]). In short, a propositional knowledge base Σ is causally independent w.r.t. a causal structure $\mathcal{G}$ (being a directed acyclic graph on a subset of $Var(\Sigma)$ where edges represent direct causal relations) iff (i) Σ is satisfiable and (ii) for any state S of the variables not concerned by the causal structure, and for each variable v of $\mathcal{G}$, then v is independent of its non-effects given its direct causes w.r.t. $\Sigma \wedge S$, where the non-effects of v are the

[4]By the way, let us mention that [15] introduces a notion of relevance $RX_\Sigma(X, \Phi)$ of a formula Φ to a subject matter X w.r.t. a KB Σ that can be characterized as well using abductive terms. Indeed, $RX_\Sigma(X, \Phi)$ holds iff there exists a minimal abductive explanation of Φ w.r.t. Σ that contains some variable from X. Accordingly, $RX_\Sigma(X, \Phi)$ holds iff at least one literal built up from X is relevant for minimally explaining Φ. The Σ_2^P-completeness of this additional form of relevance then comes from [10]. This completes the complexity results given in [15].

$X \sim_{\Sigma}^{Z} Y$	any Z	$Z = \emptyset$	$Z = V \setminus (X \cup Y)$
any X, Y	Π_2^{P}-complete	Π_2^{P}-complete	coNP-complete
$X = \{x\}$ or $Y = \{y\}$	Π_2^{P}-complete	Π_2^{P}-complete	coNP-complete
$X = \{x\}$ and $Y = \{y\}$	Π_2^{P}-complete	coBH$_2$-complete	coNP-complete
$X \cup Y = V$	coNP-complete	coNP-complete	coNP-complete

Table 1: complexity of conditional independence.

vertices in $\mathcal{G}$ which are neither parents nor descendents of v.

Proposition 23
Darwiche and Pearl's CAUSAL INDEPENDENCE *is* Π_2^{P}-*complete.*

5 Definability

Let us now focus on a form of dependence between sets of variables w.r.t. a KB, called *definability*. Definability is a strong form of dependence: while (conditionally) dependent variables interact *in some situations* ω_Z, definability imposes that some variables are fixed *whenever* some other variables are fixed as well.

Definition 10 (defining families, bases)
Let $X, Y \subseteq PS$ *and* $y \in PS$.

- X **defines** y *w.r.t.* Σ*, denoted* $X \sqsubseteq_{\Sigma} y$*, iff* $\forall \omega_X \in \Omega_X, \omega_X \wedge \Sigma \models y$ *or* $\omega_X \wedge \Sigma \models \neg y$.
- X **defines** Y *w.r.t.* Σ *(denoted* $X \sqsubseteq_{\Sigma} Y$*) iff* $\forall y \in Y, X \sqsubseteq_{\Sigma} y$.
- X **defines minimally** Y *w.r.t.* Σ *iff* $X \sqsubseteq_{\Sigma} Y$ *and no proper subset of* X *does it.*
- X *is a* **basis** *for* Y *w.r.t.* Σ *iff* X *defines minimally* Y *w.r.t.* Σ *and* Σ *is consistent.*

For instance, let b stand for "bissextile year", and $d4$ (resp. $d25$, $d100$, $d400$) for "divisible by 4" (resp. by 25, 100, by 400). The knowledge base is
$\Sigma = \{d400 \Rightarrow b, d100 \wedge \neg d400 \Rightarrow \neg b, d4 \wedge \neg d100 \Rightarrow b, \neg d4 \Rightarrow \neg b, d100 \Leftrightarrow (d4 \wedge d25)\}$.
$\{d4, d100, d400\}$ and $\{d4, d25, d400\}$ are bases for b;
$\{d25, d100, d400\}$ does not define b, because the joint falsity of these three variables does not enable telling whether b is true or false, since we do not know whether $d4$ holds or not.
$\{b, d100, d400\}$ defines $d4$, but not minimally, since $\{b, d100\}$ is a basis for $d4$; the latter also is a basis for $\{d4, d100, d400\}$.

Clearly enough, definability satisfies the following properties:

Proposition 24
(1) $\sqsubseteq_{\Sigma}$ *is transitive.*
(2) If $X' \subseteq X$*, then* $X \sqsubseteq_{\Sigma} X'$.
(3) If $X \sqsubseteq_{\Sigma} Y$ *and* $X \sqsubseteq_{\Sigma} Y'$*, then* $X \sqsubseteq_{\Sigma} Y \cup Y'$.
(4) If $X \sqsubseteq_{\Sigma} Y$ *and* $\Sigma' \models \Sigma$*, then* $X \sqsubseteq_{\Sigma'} Y$.

Easy consequences of such properties are:
(5) $X \sqsubseteq_{\Sigma} X$.
(6) If $X \sqsubseteq_{\Sigma} Y$ and $X' \sqsubseteq_{\Sigma} Y'$, then $X \cup X' \sqsubseteq_{\Sigma} Y \cup Y'$.

As a corollary of Beth's theorem [3] (stated in the more general framework of first-order logic), we get the equivalence between this *implicit* form of definability and the following *explicit* form: X (explicitly) defines y w.r.t. Σ iff there exists a formula Φ_y s.t. $Var(\Phi_y) \subseteq X$ and $\Sigma \models (\Phi_y \Leftrightarrow y)$; in this case, Φ_y is clearly unique up to equivalence modulo Σ.

Going back to our example, we get the following explicit definitions:
$\Sigma \models (d4 \wedge (\neg d100 \vee d400)) \Leftrightarrow b$;
$\Sigma \models (d4 \wedge (\neg d25 \vee d400)) \Leftrightarrow b$;
$\Sigma \models (b \vee d100) \Leftrightarrow d4$;
$\Sigma \models (d100 \wedge b) \Leftrightarrow d400$;

The following equivalent formulation is due to Padoa[23]. It is useful for looking for tractable restrictions of definability.

Proposition 25 (Padoa's method) *[23]*
For any Σ *and any* $X \subseteq PS$*, let* $rename(\Sigma, X)$ *be the formula obtained by replacing in* Σ *in a uniform way every variable from* $Var(\Sigma) \setminus X$ *by a new variable. If* $y \notin X$*, then (*$X \sqsubseteq_{\Sigma} y$ *iff* $(\Sigma \wedge rename(\Sigma, X)) \models (y \Leftrightarrow rename(y, \emptyset))$*).*

Clearly enough, whenever y does not belong to X, checking definability comes down to a standard deduction check. Since $X \sqsubseteq_{\Sigma} y$ trivially holds in the remaining case (i.e. $y \in X$), we can conclude that a set membership test plus a deduction check are always sufficient to decide definability.

Interestingly, the previous notion of variable definability can be easily turned into a notion of formula definability. Formally:

Definition 11 *X defines Ψ w.r.t. Σ (noted $X \sqsubseteq_\Sigma \Psi$) iff $\forall \omega_X \in \Omega_X, \omega_X \models_\Sigma \Psi$ or $\omega_X \models_\Sigma \neg\Psi$. $Defin(X, \Sigma) = \{\Psi \mid X$ defines Ψ w.r.t. $\Sigma\}$ is the sublanguage implicitely defined by X w.r.t. Σ.*

$Defin(X, \Sigma)$ is the set of formulas Ψ the truth values of which are known whenever the truth values of variables of X are known. *Defin* exhibits the following metatheoretic properties:

Proposition 26
(1) If $X \sqsubseteq_\Sigma Y$, then $Defin(Y, \Sigma) \subseteq Defin(X, \Sigma)$.
(2) If $\phi, \psi \in Defin(X, \Sigma)$, then $\phi \wedge \psi \in Defin(X, \Sigma)$, $\phi \vee \psi \in Defin(X, \Sigma)$, $\neg\phi \in Defin(X, \Sigma)$.

While formula definability extends variable definability (since every variable y can be also viewed as the formula y), it can be recovered from it easily:

Proposition 27 *Let new be a variable of $PS \setminus Var(\Psi)$. $X \sqsubseteq_\Sigma \Psi$ iff $X \sqsubseteq_{\Sigma \wedge (\Psi \Leftrightarrow new)} new$.*

This proposition shows how an algorithm for checking variable definability can be used to decide formula definability.

Definability can be used to discriminate variables within a knowledge base in many ways, both as defining as well as defined variables. The most important ones are the variables which are necessary for defining Y, i.e., those which belong to all bases for Y. Necessary variables will be called *key variables* by analogy with the terminology used in the database community. When designing minimal tests, key variables correspond to necessary tests, which will have to be performed anyway (and thus, when building up conditional test plans, these necessary tests can be performed at once and in parallel without needing for any outcome). When a key variable cannot be tested, the set of hypotheses (diagnoses, etc.) cannot be fully discriminated. Computing key variables in a preliminary step can also prove valuable for improving the computation of the set of all bases for Y w.r.t. Σ.

Definition 12
x is a key variable *for Y w.r.t. Σ if and only if x belongs to all bases of Y w.r.t. Σ.*

Going back to our example, given that $Var(\Sigma) = \{b, d4, d25, d100, d400\}$, $d4$ and $d400$ are the only key variables for b; $d100$ is not a key variable for b (because $\{d4, d25, d400\}$ defines b and does not contain $d100$). The key variables for $Var(\Sigma)$ are $d4$ and $d400$. Lastly, $d25$ is the unique key variable for $d25$, since $d25$ is the unique basis for $d25$ w.r.t. Σ. As this example illustrates it, all the variables y of Y that cannot be defined (except in a trivial way) in Σ are key variables for Y.

Definition 13 *y is undefinable in Σ iff for all $X \subseteq Var(\Sigma)$, if $X \sqsubseteq_\Sigma y$, then $y \in X$.*

Now, since $Y \sqsubseteq_\Sigma Y$ trivially holds, it is easy to show that:

Proposition 28 *x is a key variable for Y w.r.t. Σ iff $x \in Y$ and x is undefinable in Σ.*

Moreover, since $\sqsubseteq_\Sigma$ is a transitive relation, undefinability in Σ can be considered as a specific case of non-definability.

Proposition 29 *y is undefinable in Σ iff $Var(\Sigma) \setminus \{y\} \not\sqsubseteq_\Sigma y$.*

Undefinability in Σ also has the following characterizations that prove helpful for deriving algorithms for deciding it.

Proposition 30 *The next three statements are equivalent:*
(1) y is undefinable in Σ;
(2) $\Sigma_{y \leftarrow 0} \wedge \Sigma_{y \leftarrow 1}$ is satisfiable.
(3) There exists a prime implicant of Σ that does not contain y nor $\neg y$.

Another way to simplify the search for defining families for a given variable y is to take account of some prior knowledge concerning independence. Intuitively, a basis for y is expected not to contain variables independent of y (the notion of independence we need here is the *ceteris paribus* one).

Proposition 31 *If $x \sim_\Sigma^{Var(\Sigma) \setminus \{x,y\}} y$ and X is a basis for y, then $y \notin X$.*

The following remark is not especially helpful from a computational perspective but shows a close connection between the definitions of conditional independence and definability. Notice that, although the original definition of conditional independence between X and Y w.r.t. Z requires X and Y to be disjoint, we may wish to apply the definition in the case where X and Y intersect. For $X = Y$, it leads to $X \sim_\Sigma^Z X$ if and only if $Z \sqsubseteq_\Sigma X$, which proves that definability is covered by (unrestricted) conditional independence. More generally, $X \sim_\Sigma^Z Y$ iff $Z \sqsubseteq_\Sigma X \cap Y$ and $X \setminus Y \sim_\Sigma^Z Y \setminus X$.

We now give the complexity of definability in the general case, as well as in some restricted cases.

Proposition 32 (complexity of definability)
The results are synthesized on Table 2.

Since the transformation from formula definability to variable definability given by Proposition 27 can be achieved in polynomial time and since variable definability is a restriction of formula definability, these complexity results apply as well to formula definability.

Some tractable restrictions can be derived from Proposition 25.

Proposition 33 *Whenever Σ belongs to a class $\mathcal{C}$ of formulas that is tractable for clausal query answering and stable for expansion with partial renaming (i.e. if Σ belongs to $\mathcal{C}$ and X is a set of variables, then $\Sigma \wedge rename(\Sigma, X)$ belongs to $\mathcal{C}$ as well), then the four above problems are in* P.

Accordingly, when Σ is restricted to a renamable Horn formula or a set of binary clauses, then BASIS CHECKING becomes tractable. This is also the case when Σ is in Disjunctive Normal Form.

Proposition 34 *Whenever Σ is a DNF formula, then all four definability problems are in* P.

More interestingly, when Σ belongs to one of the above classes, the *function problem* associated with BASIS CHECKING is in FP, i.e. computing a basis for Y w.r.t. Σ can be achieved in time polynomial in $|\Sigma| + |Y|$. Additionally, when such tractable restrictions are considered and X is a basis for Y w.r.t. Σ, the truth value of every variable y of Y can be computed in time polynomial in $|\Sigma| + |X|$ for every X-world ω_X. Indeed, when Σ is tractable for clausal query answering, cheking whether $\Sigma \models (\omega_X \Rightarrow y)$ holds can be done in polynomial time and the truth value of this test gives the truth value of y.

We have also derived the following related complexity results:

Proposition 35
(1) Deciding whether y is undefinable in Σ is NP-*complete.*
(2) Deciding whether x is a key variable for Y w.r.t. Σ is NP-*complete.*

6 Taking advantage of independence/dependence

The main reason why the results presented in this paper are significant is that independence/dependence (under its various forms) is central in knowledge representation and more generally in artificial intelligence [1]. Due to space limitations, we present in the following only some of the possible uses of the notions of independence/dependence discussed in the paper.

First, it is noticeable that in many works, the independence relation between propositional variables is explicit (part of the input), while ours is implicit (derived from the input). Thus, computing independence relations from a knowledge base can be seen as an upstream task enabling us specifying the "core" (minimal) independence relation; this core independence relation can then be completed by specifying explicitely some additional dependencies using knowledge about the domain. Such an approach has been proposed in [19], where the well-known *possible models approach* update operator is recovered by using the above principle with a specific implicit dependence relation (*novelty*, see [13] [18] for details). Using in a similar way the various implicit dependence relations given in Sections 3 and 4 should lead to a variety of belief update operators. This topic is left for further research.

Definability also has many applications to knowledge representation and reasoning. In a companion paper [16], the practical computation of defining families is investigated; we also introduce a closely related concept (however different), namely *hypothesis discrimination*, and discuss its role in various fields such as model-based diagnosis or decision under uncertainty.

Beyond independence and definability, there are other types of relations between propositional variables which are worth investigating. For instance, Boutilier [4] defines the notions of *controllability* and *influenceability* which are related to the notions of independence and definability (characterizations by means of prime implicates hold as well [4]). Influenceability, which appears to be strongly connected to formula-variable independence, is briefly evoked in Section 3. Complexity results for controllability are given in [16].

7 Conclusion

This paper has brought some formal contributions to the notion of logical independence, and has then defined and studied a new form of dependence, namely, definability. As to independence, several notions have been investigated. Some of which were already known from the works of Lakemeyer, Darwiche and Pearl, Boutilier, and others are new proposals. For each notion, characterizations results have been given – many of which make use of prime implicants / implicates – that helped us pointing out some connections with ab-

definability	without minimality	with minimality
without Σ satisfiable	coNP-complete	BH_2-complete
with Σ satisfiable	BH_2-complete	BH_2-complete

Table 2: complexity of definability.

duction, especially with the works of Eiter and Gottlob [10]. Moreover, the complexity of the different notions encountered has been analyzed. As to definability, several characterization results have been provided and complexity issues have been investigated.

Our results call for several remarks and further questions. First, the complexity results established tend to confirm that checking whether some (in)dependence relation holds is generally very hard, except in some restricted cases which are tractable. Formula-variable independence, which could appear at a first glance as an easy problem, is already coNP-complete. Conditional independence is even higher, since most problems are located at the second level of the polynomial hierarchy.

In this paper independence and definability notions were induced directly from the (purely propositional) knowledge base. Further work will explore the connections not only with probabilistic independence but also with notions based on qualitative uncertainty calculi [26] [9] or on utility-theoretic notions, and also with Bacchus and Grove's [2] notions of preferential independence. Roughly speaking, in [2] a set of variables X is preferentially independent of Y when one's preference order among X-worlds does not depend on the assignments of the variables of Y, which seems to have some similarities with some of the notions studied in this paper. Knowing that there is a growing interest for the logical representation of preferences, it would be worth studying whether logical independence relations can be satisfactorily extended in order to take account of such qualitative preferences.

Acknowledgements

The publication of this paper was strongly dependent of Andreas Herzig since he pointed out a flaw in a preliminary version. We thank him a lot. The second author has been supported in part by the IUT de Lens and a "Contrat d'objectifs de la région Nord/Pas-de-Calais".

References

[1] *Artificial Intelligence* 97(1&2), special issue on relevance, December 1997.

[2] Fahiem Bacchus and Adam J. Grove. Utility independence in a qualitative decision theory. In *Proceedings of the 5th International Conference on Principles of Knowledge Representation and Reasoning (KR'96)*, Boston, 1996, 542-552.

[3] Evert W. Beth. On Padoa's method in the theory of definition. *Indagationes mathematicae* 15, 1953, 330-339.

[4] Craig Boutilier. Toward a logic for qualitative decision theory. In *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning (KR'94)*, Bonn, 1994, 75-86.

[5] Adnan Darwiche. A logical notion of conditional independence. *Proceedings of the AAAI Fall Symposium on Relevance*, New Orleans, 1994, 36-40

[6] Adnan Darwiche. Model-based diagnosis using causal networks. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, Montreal, 1995, 211-217.

[7] Adnan Darwiche. A logical notion of conditional independence: properties and applications. *Artificial Intelligence* 97(1&2), 1997, 45-82.

[8] Adnan Darwiche and Judea Pearl. Symbolic causal networks. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI'94)*, Seattle (WA), 1994, 238-244.

[9] Didier Dubois, Luis Fariñas del Cerro, Andreas Herzig and Henri Prade. Qualitative relevance and independence: a roadmap. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97)*, Nagoya, 1997, 62-67.

[10] Thomas Eiter and Georg Gottlob. The complexity of logic-based abduction. *Journal of the ACM*, 42 (1), 1995, 3-42.

[11] Luis Fariñas del Cerro and Andreas Herzig. Belief change and dependence. In *Proceedings of the 6th Conference on Theoretical Aspects of Reasoning about Knowledge (TARK'96)*, De Zeeuwse Stromen, 1996, 147-161.

[12] Peter Gärdenfors. Belief revision and irrelevance. *PSA*, 2:349-356.

[13] Russell Greiner and Michael R. Genesereth. What's new? A semantic definition of novelty. In *Proceedings of the* 8th *International Joint Conference on Artificial Intelligence (IJCAI'83)*, Karlsruhe, 1983, 450–454.

[14] Gerhard Lakemeyer. A logical account of relevance. In *Proceedings of the* 14th *International Joint Conference on Artificial Intelligence (IJCAI'95)*, Montreal, 1995, 853-859.

[15] Gerhard Lakemeyer. Relevance from an epistemic perspective. *Artificial Intelligence* 97(1–2), 1997, 137–167.

[16] Jérôme Lang and Pierre Marquis. Two notions of dependence in propositional logic: controllability and definability. *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI'98)*, Madison (WI), 1998.

[17] Fangzhen Lin and Ray Reiter. Forget it! In *Proceedings of the AAAI Fall Symposium on Relevance*, New Orleans, 1994, 154-159.

[18] Pierre Marquis. Novelty revisited. In *Proceedings of the* 6th *International Symposium on Methodologies for Intelligent Systems (ISMIS'91)*, Charlotte(NC), Lecture Notes in Artificial Intelligence 542, Springer-Verlag, 1991, pp. 550-559.

[19] Pierre Marquis. Possible models approach via independency. In *Proceedings of the* 11th *European Conference on Artificial Intelligence (ECAI'94)*, Amsterdam, 1994, 336-340.

[20] A. Padoa. Essai d'une théorie algébrique des nombres entiers, précédé d'une introduction logique à une théorie déductive quelconque. In *Bibliothèque du Congrès International de Philosophie*, Paris, 1903, 309-365.

[21] Christos H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[22] Ray Reiter and Johan de Kleer. Foundations of assumption-based truth maintenance systems: preliminary report. In *Proceedings of the* 6th *National Conference on Artificial Intelligence (AAAI'87)*, Seattle (WA), 1987, 183–188.

[23] A. Padoa. Essai d'une théorie algébrique des nombres entiers, précédé d'une introduction logique à une théorie déductive quelconque. In *Bibliothèque du Congrès International de Philosophie*, Paris, 1903, 309-365.

[24] Christos H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[25] Ray Reiter and Johan de Kleer. Foundations of assumption-based truth maintenance systems: preliminary report. In *Proceedings of the* 6th *National Conference on Artificial Intelligence (AAAI'87)*, Seattle (WA), 1987, 183–188.

[26] Milan Studený. Formal properties of conditional independence in different calculi of artificial intelligence. In *Proceedings of the European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, Granada, Lecture Notes in Computer Science 747, Springer-Verlag, 1993, 341-348.

The Complexity of Model Checking in Modal Event Calculi with Quantifiers

Iliano Cervesato
Department of Computer Science
Stanford University
Stanford, CA 94305-9045
iliano@cs.stanford.edu

Massimo Franceschet **Angelo Montanari**
Dipartimento di Matematica e Informatica
Università di Udine
Via delle Scienze, 206 - 33100 Udine, Italy
{franzesc|montana}@dimi.uniud.it

Abstract

Kowalski and Sergot's Event Calculus (*EC*) is a simple temporal formalism that, given a set of event occurrences, derives the maximal validity intervals (MVIs) over which properties initiated or terminated by these events hold. It does so in polynomial time with respect to the number of events. Extensions of its query language with Boolean connectives and operators from modal logic have been shown to improve substantially its scarce expressiveness, although at the cost of an increase in computational complexity. However, significant sublanguages are still tractable. In this paper, we further extend *EC* queries by admitting arbitrary event quantification. We demonstrate the added expressive power by encoding a hardware diagnosis problem in the resulting calculus. We conduct a detailed complexity analysis of this formalism and several sublanguages that restrict the way modalities, connectives, and quantifiers can be interleaved. We also describe an implementation in the higher-order logic programming language λ*Prolog*.

1 Introduction

The *Event Calculus*, abbreviated *EC* [9], is a simple temporal formalism designed to model and reason about scenarios characterized by a set of *events*, whose occurrences have the effect of starting or terminating the validity of determined properties. Given a *possibly incomplete* description of when these events take place and of the properties they affect, *EC* is able to determine the *maximal validity intervals*, or *MVIs*, over which a property holds uninterruptedly. In practice, since this formalism is usually implemented as a logic program, *EC* can also be used to check the truth of *MVIs* and process boolean combinations of *MVI* verification or computation requests. The range of queries that can be expressed in this way is however too limited for modeling realistic situations.

A systematic analysis of *EC* has recently been undertaken in order to gain a better understanding of this calculus and determine ways of augmenting its expressive power. The keystone of this endeavor has been the definition of an extendible formal specification of the functionalities of this formalism [3]. This has had the effects of establishing a semantic reference against which to verify the correctness of implementations [4], of casting *EC* as a model checking problem [5], and of setting the ground for studying the complexity of this problem, which was proved polynomial [2]. Extensions of this model have been designed to accommodate constructs intended to enhance the expressiveness of *EC*. In particular, modal versions of *EC* [1], the interaction between modalities and connectives [5], and preconditions [6] have all been investigated in this context.

In this paper, we continue this endeavor to enhance the expressive power of *EC* by considering the possibility of quantifying over events in queries, in conjunction with boolean connectives and modal operators. We also admit requests to check the relative order of two events. We thoroughly analyze the representational and computational features of the resulting formalism, that we call *QCMEC*. We also consider two proper sublanguages of it, *EQCMEC*, in which modalities are applied to atomic formulas only, and *CMEC*, which is quantifier-free. We show that *QCMEC* and its restrictions can effectively be used to encode diagnosis problems. Moreover, we provide an elegant implementation in the higher-order logic programming language λ*Prolog* [10] and prove its soundness and completeness. As far as computational complexity is concerned, we prove that model checking in *CMEC*, *EQCMEC*, and *QCMEC* is **PSPACE**-complete. However, while solving an *EQCMEC* problem is exponential in the size of the query, it has only polynomial cost in the number n of events, thus making *EQCMEC* a viable formalism for *MVI* verification or computation. Since in most realistic applications the size of databases (n) dominates by several orders of magnitude the size of the query, n is asymptotically the parameter of interest.

The main contributions of this work are: (1) the extension of a family of modal event calculi with quantifiers; (2) permitting queries to mention ordering information; (3) the use of the higher-order features of modern logic programming languages in temporal reasoning; and (4) analyzing the complexity of model checking in these extensions of *EC*.

This paper is organized as follows. In Section 2, we formalize *QCMEC* and significant subcalculi. Section 3 exemplifies how this calculus can adequately model certain hardware diagnosis problems. In Section 4, we briefly introduce the logic programming language *λProlog*, give an implementation of *QCMEC* in it and prove the soundness and completeness of the resulting program. We study the complexity of *QCMEC* and its sublanguages in Section 5. We outline directions of future work in Section 6.

2 Modal Event Calculi with Quantifiers

In this section, we first briefly recall the syntax and semantics of a number of modal event calculi. We invite the interested reader to consult [1, 3, 5, 8, 9] for motivations, examples, properties, and technical details. We then extend these basic definitions to give a semantic foundation to refinements of these calculi with quantifiers.

2.1 Event Calculus

The Event Calculus (*EC*) [9] and the extensions we propose aim at modeling scenarios that consist of a set of events, whose occurrences over time have the effect of initiating or terminating the validity of properties, some of which may be mutually exclusive. We formalize the time-independent aspects of a situation by means of an *EC-structure* [1], defined as follows:

Definition 2.1 (*EC-structure*)

A structure *for the* Event Calculus (*or* EC-structure) *is a quintuple* $\mathcal{H} = (E,\ P,\ [\cdot\rangle,\ \langle\cdot],\ ]\cdot,\cdot[)$ *such that:*

- $E = \{e_1, \ldots, e_n\}$ *and* $P = \{p_1, \ldots, p_m\}$ *are finite sets of* events *and* properties, *respectively.*
- $[\cdot\rangle : P \rightarrow \mathbf{2}^E$ *and* $\langle\cdot] : P \rightarrow \mathbf{2}^E$ *are respectively the* initiating *and* terminating map *of* $\mathcal{H}$. *For every property* $p \in P$, $[p\rangle$ *and* $\langle p]$ *represent the set of events that initiate and terminate* p*, respectively.*
- $]\cdot,\cdot[\subseteq P \times P$ *is an irreflexive and symmetric relation, called the* exclusivity relation, *that models exclusivity among properties.* □

The temporal aspect of *EC* is given by the order in which events happen. Unlike the original presentation [9], we focus our attention on situations where the occurrence time of events is unknown and only assume the availability of incomplete information about the relative order in which they have happened. We however require the temporal data to be consistent so that an event cannot both precede and follow some other event. Therefore, we formalize the time-dependent aspect of a scenario modeled by *EC* by means of a (strict) *partial order*, i.e. an irreflexive and transitive relation, over the involved set of event occurrences. We write $W_{\mathcal{H}}$ for the set of all partial orders over the set of events E in an *EC*-structure $\mathcal{H}$, use the letter w to denote individual orderings, or *knowledge states*, and write $e_1 <_w e_2$ to indicate that e_1 precedes e_2 in w. The set $W_{\mathcal{H}}$ of all knowledge states naturally becomes a reflexive ordered set when considered together with the usual subset relation $\subseteq$, which is indeed reflexive, transitive and antisymmetric. An *extension* of a knowledge state w is any element of $W_{\mathcal{H}}$ that contains w as a subset. We write $\text{Ext}_{\mathcal{H}}(w)$ for the set of all extensions of the ordering w in $W_{\mathcal{H}}$.

Given a structure $\mathcal{H} = (E,\ P,\ [\cdot\rangle,\ \langle\cdot],\ ]\cdot,\cdot[)$ and a knowledge state w, *EC* permits inferring the *maximal validity intervals*, or *MVIs*, over which a property p holds uninterruptedly. We represent an *MVI* for p as $p(e_i, e_t)$, where e_i and e_t are the events that respectively initiate and terminate the interval over which p holds maximally. Consequently, we adopt as the *query language* of *EC* the set $\mathcal{L}_{\mathcal{H}}(EC) = \{p(e_1, e_2) : p \in P \text{ and } e_1, e_2 \in E\}$ of all such property-labeled intervals over $\mathcal{H}$. We interpret the elements of $\mathcal{L}_{\mathcal{H}}(EC)$ as propositional letters and the task performed by *EC* reduces to deciding which of these formulas are MVIs in the current knowledge state w and which are not. This is a model checking problem.

In order for $p(e_1, e_2)$ to be an MVI relative to the event ordering w, it must be the case that $e_1 <_w e_2$. Moreover, e_1 and e_2 must witness the validity of the property p at the ends of this interval by initiating and terminating p, respectively. The maximality requirement is caught by the negation of the meta-predicate $br(p, e_1, e_2, w)$ below, which expresses the fact that the validity of an MVI must not be *broken* by any interrupting event. Any event e which is known to have happened between e_1 and e_2 in w and that initiates or terminates a property that is either p itself or a property exclusive with p interrupts the validity of $p(e_1, e_2)$ [4].

Definition 2.2 (*Intended model of EC*)

Let $\mathcal{H} = (E,\ P,\ [\cdot\rangle,\ \langle\cdot],\ ]\cdot,\cdot[)$ be a EC-structure. The intended EC-model *of $\mathcal{H}$ is the propositional valuation $\upsilon_{\mathcal{H}} : W_{\mathcal{H}} \to 2^{\mathcal{L}_{\mathcal{H}}(EC)}$, where $p(e_1, e_2) \in \upsilon_{\mathcal{H}}(w)$ if and only if* (i) $e_1 <_w e_2$, (ii) $e_1 \in [p\rangle$, (iii) $e_2 \in \langle p]$, (iv) $br(p, e_1, e_2, w)$ *does not hold, where $br(p, e_1, e_2, w)$ abbreviates*

> *there exists an event $e \in E$ such that $e_1 <_w e$, $e <_w e_2$ and there exists a property $q \in P$ such that $e \in [q\rangle$ or $e \in \langle q]$, and either $]p, q[$ or $p = q$.* □

2.2 Modal EC with Connectives

The query language of the basic EC we just formalized suffers from a remarkably low expressive power that prevents its use for modelling any but the most trivial applications. The expressiveness of this formalism is drammatically augmented by admitting boolean connectives in queries. This allows inquiring about logical combinations of basic MVI verification problems.

In our specific setting, where the ordering of event occurrences is only partially specified, the set of MVIs computed by EC is not stable with respect to the acquisition of new ordering information. Indeed, as we move to an extension of the current knowledge state, some MVIs might become invalid and new MVIs can emerge [7]. Extending the query language of EC with the modal logic operators $\Box$ and $\Diamond$ leads to the possibility of enquiring about which MVIs will remain valid in every extension of the current knowledge state, and about which intervals might become MVIs in some extension of it [1, 8]. Several ways of combining boolean connectives and modalities, with different cost and expressiveness, have been proposed [3, 5].

In this paper, we also include a *precedence test* operator, written $<$, which allows checking the relative order of two events in the current knowledge state. In previous work, this was awkwardly achieved either by augmenting EC-structures with dedicated properties [5], or by using preconditions [6]. A native precedence test makes inquiring about the relative order of two events independent from the underlying EC-structure.

Given an EC-structure $\mathcal{H}$, the query language that freely includes these three extensions is formally defined by the following grammar:

$$\varphi \ ::= \ p(e_1, e_2) \mid e_1 < e_2 \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \\ \mid \varphi_1 \vee \varphi_2 \mid \Box\varphi \mid \Diamond\varphi.$$

We call this language $\mathcal{L}_{\mathcal{H}}(CMEC)$ and $CMEC$ the relative extension of EC. In addition to the above operators, we admit implication as a derived connective, where $\varphi_1 \supset \varphi_2$ is classically defined as $\neg\varphi_1 \vee \varphi_2$.

In order to formalize the semantics of the modalities in $CMEC$, we must shift the focus from the current knowledge state w to all knowledge states that are reachable from w, i.e. $\mathrm{Ext}_{\mathcal{H}}(w)$. Since $\subseteq$ is a reflexive partial order, $(W_{\mathcal{H}}, \subseteq)$ can be naturally viewed as a finite, reflexive, transitive and antisymmetric modal frame. If we consider this frame together with the straightforward modal extension of the valuation $\upsilon_{\mathcal{H}}$ to an arbitrary knowledge state, we obtain a modal model for $CMEC$. Connectives are handled as usual and incorporating the precedence test is trivial.

Definition 2.3 (*Intended model of CMEC*)

Let $\mathcal{H} = (E,\ P,\ [\cdot\rangle,\ \langle\cdot],\ ]\cdot,\cdot[)$ be an EC-structure. The intended $CMEC$-model *of $\mathcal{H}$ is the modal model $\mathcal{I}_{\mathcal{H}} = (W_{\mathcal{H}}, \subseteq, \upsilon_{\mathcal{H}})$, where the propositional valuation $\upsilon_{\mathcal{H}} : W_{\mathcal{H}} \to 2^{\mathcal{L}_{\mathcal{H}}(EC)}$ is defined as in Definition 2.2. Given $w \in W_{\mathcal{H}}$ and $\varphi \in \mathcal{L}_{\mathcal{H}}(CMEC)$, the truth of φ at w with respect to $\mathcal{I}_{\mathcal{H}}$, denoted by $\mathcal{I}_{\mathcal{H}}; w \models \varphi$, is defined as follows:*

$\mathcal{I}_{\mathcal{H}}; w \models p(e_1, e_2)$ *iff* $p(e_1, e_2) \in \upsilon_{\mathcal{H}}(w)$;
$\mathcal{I}_{\mathcal{H}}; w \models e_1 < e_2$ *iff* $e_1 <_w e_2$;
$\mathcal{I}_{\mathcal{H}}; w \models \neg\varphi$ *iff* $\mathcal{I}_{\mathcal{H}}; w \not\models \varphi$;
$\mathcal{I}_{\mathcal{H}}; w \models \varphi_1 \wedge \varphi_2$ *iff* $\mathcal{I}_{\mathcal{H}}; w \models \varphi_1$ *and* $\mathcal{I}_{\mathcal{H}}; w \models \varphi_2$;
$\mathcal{I}_{\mathcal{H}}; w \models \varphi_1 \vee \varphi_2$ *iff* $\mathcal{I}_{\mathcal{H}}; w \models \varphi_1$ *or* $\mathcal{I}_{\mathcal{H}}; w \models \varphi_2$;
$\mathcal{I}_{\mathcal{H}}; w \models \Box\varphi$ *iff for all* $w' \in \mathrm{Ext}_{\mathcal{H}}(w)$, $\mathcal{I}_{\mathcal{H}}; w' \models \varphi$;
$\mathcal{I}_{\mathcal{H}}; w \models \Diamond\varphi$ *iff there is* $w' \in \mathrm{Ext}_{\mathcal{H}}(w)$ *such that* $\mathcal{I}_{\mathcal{H}}; w' \models \varphi$. □

Notice that deciding the truth of a modal formula requires the exploration of all the extensions of the current knowledge state. Since there are exponentially many, this raises the complexity of $CMEC$ beyond tractability [5]. This distressing fact is overcome in the calculus $ECMEC$ [4, 5], that restricts $CMEC$ by allowing $\Box$ and $\Diamond$ to enclose only atomic formulas of the form $e_1 < e_2$ and $p(e_1, e_2)$. To determine the truth of atomic formulas prefixed by one modal operator, it is possible to exploit necessary and sufficient *local conditions* over the given partial order, thus avoiding a complete (and expensive) search of all the consistent extensions of the given order [5]. Therefore, solving modal queries in $ECMEC$ has polynomial cost [5].

This is particularly appealing since numerous $CMEC$-formulas are logically equivalent to $ECMEC$-formulas. The transformation proceeds by pushing the modalities inside the scope of the connectives. An $ECMEC$ formula cannot always be produced since $\Box$ does not distribute over $\vee$, and dually $\diamond$ cannot be pushed inside a conjunction. We will now consider conditions that permit overcoming this difficulty in situations of interest.

Specifically, we consider EC-structures $\mathcal{H} = (E,\ P,\ [\cdot\rangle,\ \langle\cdot],\ ]\cdot,\cdot[)$ where every property is initiated and terminated by at most one event and there are no exclusive properties. We call this condition $(*)$. An atomic formula $p(e_1, e_2)$ on $\mathcal{H}$ is an MVI relative to the knowledge state $w \in W_{\mathcal{H}}$ if and only if e_1 initiates p, e_2 terminate p and (e_1, e_2) belongs to w. Indeed, condition $(*)$ ensures us that there are no interrupting events for p in (e_1, e_2) and thus we do not need to check whether $br(e_1, p, e_2, w)$ holds since this meta-predicate will be trivially false. Condition $(*)$ offers further opportunities to push modalities inside the scope of connectives. We omit the proof of the following simple proposition.

Proposition 2.4 (*Consequences of* $(*)$)

Let $\mathcal{H} = (E,\ P,\ [\cdot\rangle,\ \langle\cdot],\ ]\cdot,\cdot[)$ be an EC-structure that satisfies $(*)$. Let φ be a CMEC*-formula. For $p \in P$ and $e_1, e_2 \in E$, let $\nu_p(e_1, e_2)$ be either $p(e_1, e_2)$ or $(e_1 < e_2)$. Then, for any $w \in W_{\mathcal{H}}$ such that $e_1 <_w e_2$, we have that:*

i. $w \models \Box(\nu_p(e_1, e_2) \vee \varphi)$ *iff* $w \models \nu_p(e_1, e_2) \vee \Box\varphi$;
ii. $w \models \diamond(\nu_p(e_1, e_2) \wedge \varphi)$ *iff* $w \models \nu_p(e_1, e_2) \wedge \diamond\varphi$. ■

In particular, for $\varphi =$ **false** (resp. **true**), we have that $w \models \Box\nu_p(e_1, e_2)$ (resp. $w \models \diamond\nu_p(e_1, e_2)$) *iff* $w \models \nu_p(e_1, e_2)$.

2.3 Modal EC with Connectives and Quantifiers

We will now enrich *CMEC* with explicit universal and existential event quantifiers that can be used freely in a query. We call the resulting formalism *QCMEC*. Indeed, a logic programming implementation of *CMEC* can emulate only restricted forms of existential quantification by means of unification, while universally quantified queries are out of reach.

In order to accommodate quantifiers, we extend the query language of an *EC*-structure $\mathcal{H} = (E,\ P,\ [\cdot\rangle,\ \langle\cdot],\ ]\cdot,\cdot[)$ in several respects. We first assume the existence of infinitely many *event variables* that we denote x, possibly subscripted. We write $\bar{e}$ for a syntactic entity that is either an event in E or an event variable. The query language of *QCMEC*, denoted $\mathcal{L}_{\mathcal{H}}(QCMEC)$, is the set of *closed* formulas generated by the following grammar:

$$\begin{aligned}\varphi \ ::= \ & p(\bar{e}_1, \bar{e}_2) \mid \bar{e}_1 < \bar{e}_2 \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \\ \mid \ & \varphi_1 \vee \varphi_2 \mid \Box\varphi \mid \diamond\varphi \mid \forall x.\,\varphi \mid \exists x.\,\varphi.\end{aligned}$$

The notions of free and bound variables are defined as usual and we identify formulas that differ only by the name of their bound variables. We write $[e/x]\varphi$ for the substitution of an event $e \in E$ for every free occurrence of the event variable x in the formula φ. Notice that this limited form of substitution cannot lead to variable capture.

We now extend the notion of intended model to accommodate quantifiers.

Definition 2.5 (*Intended model of QCMEC*)

Let $\mathcal{H} = (E,\ P,\ [\cdot\rangle,\ \langle\cdot],\ ]\cdot,\cdot[)$ be an EC-structure. The intended *QCMEC*-*model of $\mathcal{H}$ is the modal model $\mathcal{I}_{\mathcal{H}} = (W_{\mathcal{H}}, \subseteq, v_{\mathcal{H}})$ defined as in Definition 2.3. Given $w \in W_{\mathcal{H}}$ and a (closed) formula $\varphi \in \mathcal{L}_{\mathcal{H}}(QCMEC)$, the truth of φ at w with respect to $\mathcal{I}_{\mathcal{H}}$, denoted as $\mathcal{I}_{\mathcal{H}}; w \models \varphi$, is defined as in Definition 2.3 with the addition of the following two cases:*

$\mathcal{I}_{\mathcal{H}}; w \models \forall x.\,\varphi$ *iff for all* $e \in E$, $\mathcal{I}_{\mathcal{H}}; w \models [e/x]\varphi$;
$\mathcal{I}_{\mathcal{H}}; w \models \exists x.\,\varphi$ *iff there exists* $e \in E$ *such that* $\mathcal{I}_{\mathcal{H}}; w \models [e/x]\varphi$. □

The well-foundedness of this definition derives from the observation that if $\forall x.\,\varphi$ and $\exists x.\,\varphi$ are closed formula, so is $[e/x]\varphi$ for every event $e \in E$.

A universal quantification over a finite domain can always be expanded into a finite sequence of conjunctions. Similarly an existentially quantified formula is equivalent to the disjunction of all its instances. The following lemma, whose simple proof we omit, applies these principles to *QCMEC*.

Lemma 2.6 (*Unfolding quantifiers*)

Let $\mathcal{H} = (E,\ P,\ [\cdot\rangle,\ \langle\cdot],\ ]\cdot,\cdot[)$ be an EC-structure, *with $E = \{e_1, \ldots, e_n\}$. Then, for every $w \in W_{\mathcal{H}}$,*

i. $\mathcal{I}_{\mathcal{H}}; w \models \forall x.\,\varphi$ *iff* $\mathcal{I}_{\mathcal{H}}; w \models \bigwedge_{i=1}^{n} [e_i/x]\varphi$;
ii. $\mathcal{I}_{\mathcal{H}}; w \models \exists x.\,\varphi$ *iff* $\mathcal{I}_{\mathcal{H}}; w \models \bigvee_{i=1}^{n} [e_i/x]\varphi$. ■

This property hints at the possibility of compiling a *QCMEC* query to a quantifier-free formula. Observe however that this is possible only after an *EC*-structure has been specified. We will rely on the above lemma in order to analyze the explicit complexity of the formalism in Section 5. It is also possible to take advantage of it in order to structure an implementation of *QCMEC* into a preprocessor that expands quantifiers into exhaustive sets of conjunctions or disjunctions, and a *CMEC* checker that verifies the resulting formula. We will however follow a more direct approach in Section 4.

We conclude this section by defining a quantified variant of the previously introduced formalism *ECMEC*. The calculus *EQCMEC* differs from *QCMEC* by imposing that propositional connectives and quantifiers be external to the scope of the modal operators.

3 Example

In this section, we consider a case study taken from the domain of hardware fault diagnosis that shows how an extension of *EC* with quantifiers, connectives and modalities can be conveniently used to model real-world applications.

We focus our attention on the representation and information processing of fault symptoms that are spread over periods of time and for which current expert system technology is particularly deficient [11]. Consider the following example, which diagnoses a fault in a computerized numerical control center for a production chain.

> *A possible cause for an undefined position of the tool magazine is a faulty limit switch S. This cause can however be ruled out if the status registers R_1 and R_2 show the following behavior in every session: from a situation in which both registers contain the value* 0*, they assume the value* 1 *in successive and disjoint time intervals (first R_1 and then R_2), and then return to* 0*. A session is a time interval initiated when a special register C is set to* 1 *and terminated when C is reset to* 0.

Figure 1 describes a possible sequence of transitions for R_1 and R_2 within an individual session i. If every recorded session has a similar pattern, the eventuality of S being faulty can be excluded. In order to verify this behavior, the contents of the registers must be monitored over time. Typically, each value (0 or 1) of a register persists for at least t time units. Measurements are made at fixed intervals (sampling periods), asynchronously with the change of value of the registers. In order to avoid losing register transitions, measurements must be made frequently enough, that is, the sampling period must be less than t. However, it may happen that transitions of *different* registers take place between two consecutive measurements, making it impossible to recover their relative order.

This situation is depicted in Figure 1, where dotted lines indicate measurements. Moreover, we have given names to the individual transitions of state of the different registers. In this specific situation, the values found at measurements m_0^i, m_1^i and m_2^i allow us to determine that C has acquired the value 1 and R_1 has successively been set during this interval (transitions e_0^i and e_1^i, respectively). The contents of the registers at measurement m_3^i let us infer that R_1 has been reset (transition e_2^i) and that the value of R_2 has changed to 1 (transition e_3^i). We know that both e_2^i and e_3^i have taken place after e_1^i, but we have no information about the relative order of these transitions. Finally, m_4^i and m_5^i acknowledge that R_2 has successively been reset to 0 (e_4^i), and the same has then happened to C (e_5^i).

We will now give a formalization of this example and use various modal event calculi to draw conclusions about it. The situation relative to session i depicted in Figure 1 can be represented by the *EC*-structure $\mathcal{H}^i = (E^i,\ P,\ [\cdot\rangle^i,\ \langle\cdot]^i,\ ]\cdot,\cdot[)$, whose components are defined as follows:

- $E^i = \{e_0^i, e_1^i, e_2^i, e_3^i, e_4^i, e_5^i\}$;
- $P = \{C, R_1, R_2\}$;
- $[C\rangle^i = \{e_0^i\}$, $[R_1\rangle^i = \{e_1^i\}$, $[R_2\rangle^i = \{e_3^i\}$;
- $\langle C]^i = \{e_5^i\}$, $\langle R_1]^i = \{e_2^i\}$, $\langle R_2]^i = \{e_4^i\}$;
- $]\cdot,\cdot[= \emptyset$.

We have encoded transitions as events with the same name and denoted with R_j $(j = 1, 2)$ the property that register R_j has value 1, and similarly for C.

The ordering w of the transitions inferred from the measurements corresponds to the transitive closure of the following graph.

$$\begin{tikzcd} & & e_2^i \arrow[dr] & & \\ e_0^i \arrow[r] & e_1^i \arrow[ur] \arrow[dr] & & e_4^i \arrow[r] & e_5^i \\ & & e_3^i \arrow[ur] & & \end{tikzcd}$$

Consider the formulas $\varphi^i = R_1(e_1^i, e_2^i) \ \wedge\ (e_2^i < e_3^i) \ \wedge\ R_2(e_3^i, e_4^i)$. In order to verify that the switch S is not faulty in session i, we must ensure that the registers R_1 and R_2 display the expected behavior in all refinements of the current knowledge state w. This amounts to proving that the *CMEC*-formula $\Box\varphi^i$ is true in w. If this is the case, there is no fault in session i, although other sessions might indicate that S is dysfunctional. If we want to determine the existence of at least one extension of w where the registers behave as displayed in Figure 1, we must verify the truth of $\Diamond\varphi^i$ in w. If this *CMEC*-formula is true, we cannot be sure whether S is faulty or not.

Since $\mathcal{I}_{\mathcal{H}^i}; w \models \Diamond\varphi^i$ and $\mathcal{I}_{\mathcal{H}^i}; w \not\models \Box\varphi^i$, a faulty behavior of S in session i is possible but not certain. Assume now that, unlike the actual situation depicted in Figure 1, we extend w so that e_3^i precedes e_2^i, call w_1 the resulting state. Then, $\mathcal{I}_{\mathcal{H}^i}; w_1 \not\models \Diamond\varphi^i$, that is, the evolution of the values in the registers hints at a fault. Conversely, let us refine w so that e_2^i precedes e_3^i, and call w_2 the resulting knowledge state. Then, we can infer $\mathcal{I}_{\mathcal{H}^i}; w_2 \models \Box\varphi^i$, and hence we can conclude that the switch S is certainly not faulty.

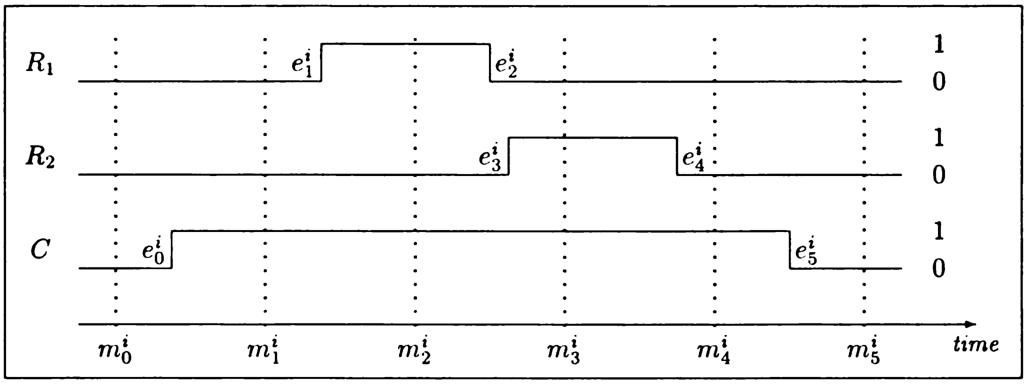


Figure 1: Expected Register Behavior and Measurements during Session i.

The formulas we have used so far belong to the language of *CMEC*. This is unfortunate since model checking is intractable in it. However, there exist equivalent formulas in the language of *ECMEC*, that can be checked in polynomial time. By the distributivity $\Box$ over $\wedge$, $\Box\varphi^i$ is equivalent to the *ECMEC*-formula:

$$\psi^i = \Box R_1(e_1^i, e_2^i) \wedge \Box(e_2^i < e_3^i) \wedge \Box R_2(e_3^i, e_4^i)$$

Therefore, we can use *ECMEC* and ψ^i to establish if the switch S is fault-free or is possibly defective.

The best approximation of $\Diamond\varphi^i$ we can achieve is the *ECMEC*-formula

$$\chi^i = \Diamond R_1(e_1^i, e_2^i) \wedge \Diamond(e_2^i < e_3^i) \wedge \Diamond R_2(e_3^i, e_4^i)$$

which, in general, is not equivalent to $\Diamond\varphi^i$. However, notice that the structure $\mathcal{H}$ satisfies condition $(*)$ (Section 2). Moreover, (e_1^i, e_2^i) and (e_3^i, e_4^i) belong to w, and thus, from Proposition 2.4, it follows that $\mathcal{I}_{\mathcal{H}^i}; w \models \chi^i$ if and only if $\mathcal{I}_{\mathcal{H}^i}; w \models \Diamond\varphi^i$. Therefore, we can use *ECMEC* and χ^i to establish whether the switch S is certainly faulty or is possibly correct.

Quantifiers allow extending this line of reasoning from session i to all recorded sessions, enabling us to give a faithful representation of the given rule for detecting faults in register S. We achieve this by using quantifiers to abstract from the specific events appearing in φ^i above. Consider the following *QCMEC*-formulas φ_1 and φ_2:

$$\begin{aligned}\alpha(\vec{x}) &= x_0 < x_1 \;\wedge\; R_1(x_1, x_2) \;\wedge \\ &\quad\; x_2 < x_3 \;\wedge\; R_2(x_3, x_4) \;\wedge\; x_4 < x_5 \\ \varphi_1 &= \forall x_0. \forall x_5. C(x_0, x_5) \supset \exists x_1. \exists x_2. \exists x_3. \exists x_4. \Box\alpha(\vec{x}) \\ \varphi_2 &= \forall x_0. \forall x_5. C(x_0, x_5) \supset \exists x_1. \exists x_2. \exists x_3. \exists x_4. \Diamond\alpha(\vec{x})\end{aligned}$$

where $\vec{x}$ stands for the list of variables $(x_0, x_1, x_2, x_3, x_4, x_5)$. Given a global *EC*-structure $\mathcal{H}$, we can be certain that the switch S is not faulty no matter how the order of actual transitions differs from what was inferred from the measurements if $\mathcal{I}_{\mathcal{H}}; w \models \varphi_1$ holds. On the other hand, the possibility that S behaves correctly is left open if $\mathcal{I}_{\mathcal{H}}; w \models \varphi_2$ is valid. Both φ_1 and φ_2 are in $\mathcal{L}_{\mathcal{H}}(QCMEC)$. However, if we distribute the modal operator over the boolean connectives in φ_1 and φ_2, we obtain two formulas, say φ_1' and φ_2', that are in the language of *EQCMEC* and thus can be solved in a time that is polynomial in the number of events (Section 5). It is possible to show that, for any $w' \in \text{Ext}(w)$, $\mathcal{I}_{\mathcal{H}}; w' \models \varphi_1$ if and only if $\mathcal{I}_{\mathcal{H}}; w' \models \varphi_1'$ and $\mathcal{I}_{\mathcal{H}}; w' \models \varphi_2$ if and only if $\mathcal{I}_{\mathcal{H}}; w' \models \varphi_2'$.

4 Implementation

The Event Calculus [9] has traditionally been implemented in the logic programming language *Prolog* [13]. Recent extensions to *EC* have instead adopted λ*Prolog* [10] in order to achieve a declarative yet simple encoding, necessary to formally establish correctness issues [4]. In this section, we will rely again on λ*Prolog* to obtain an elegant encoding of *QCMEC* and to prove its correctness. Space reasons forbid discussing the implementation of its subcalculi.

4.1 λProlog in a nutshell

Due to space limitations, we shall assume the reader to be familiar with the logic programming language *Prolog* [13]. We will instead illustrate some of the characteristic constructs of λ*Prolog* at an intuitive level. We invite the interested reader to consult [10] for a more complete discussion, and [4] for a presentation in the context of the Event Calculus.

Differently from *Prolog* which is first-order, λ*Prolog* is a ***higher-order*** language, which means that the terms in this programming language are drawn from a ***simply*** *typed λ-calculus*. More precisely, the syntax of terms is given by the following grammar:

$$M \ ::= \ c \mid x \mid F \mid M_1\, M_2 \mid x \setminus M$$

where c ranges over *constants*, x stands for a *bound variable* and F denotes a *logical variable* (akin to *Prolog*'s variables). Identifiers beginning with a lowercase and an uppercase letter stand for constants and logical variables, respectively. Terms that differ only by the name of their bound variables are considered indistinguishable. "$x \setminus M$" is *λProlog*'s syntax for *λ-abstraction*, traditionally written $\lambda x.\, M$. In this language, terms and atomic formulas are written in curried form (e.g. "`before E1 E2`" rather than "`before(E1, E2)`").

Every constant, bound variable and logical variable is given a unique type A. Types are either user-defined *base types*, or *functional types* of the form A_1 `->` A_2. By convention, the predefined base type `o` classifies formulas. A base type a is declared as "`kind` a`.`", and a constant c of type A is entered in *λProlog* as "`type` c A`.`". Syntax is provided for declaring infix symbols. Application and λ-abstraction can be typed if their subexpression satisfy certain constraints. *λProlog* will reject every term that is not typable.

While first-order terms are equal solely to themselves, the equational theory of higher-order languages identifies terms that can be rewritten to each other by means of the *β-reduction* rule: $(x \setminus M)\, N = [N/x]M$, where the latter expression denotes the capture-avoiding substitution of the term N for the bound variable x in M. A consequence of this fact is that unification in *λProlog* must perform β-reduction on the fly in order to equate terms or instantiate logical variables. A further difference from *Prolog* is that logical variables in *λProlog* can stand for functions (i.e. expressions of the form $x \setminus M$) and this must be taken into account when unification is performed.

For our purposes, the language of formulas of *λProlog* differs from *Prolog* for the availability of implication and of an explicit existential quantifier in the body of clauses. The goal $D \supset G$, written "D`=>`G" in the concrete syntax of this language, is solved by resolving the goal G after having assumed D as an additional program clause. The goal $\exists x.\, G$ is entered as "`sigma` $x \setminus G$". We will also take advantage of *negation-as-failure*, denoted `not`. We will not rely directly on the other powerful constructs offered by this language. Other connectives are denoted as in *Prolog*: "`,`" for conjunction, "`;`" for disjunction, "`:-`" for implication with the arguments reversed. The only predefined predicate we will use is the infix "`=`" that unifies its arguments. Given a well-typed *λProlog* program $\mathcal{P}$ and a goal G, the fact that there is a derivation of G from $\mathcal{P}$, i.e. that G is solvable in $\mathcal{P}$, is denoted $\mathcal{P} \vdash G$. See [4, 10] for details.

λProlog offers also the possibility of organizing programs into modules. A module m is declared as "`module` m`.`" followed by the declarations and clauses that define it. Modules can access other modules by means of the `accumulate` declaration.

Finally, `%` starts a comment that extends to the end of the line.

4.2 Implementation of QCMEC in λProlog

We will now give an implementation of *QCMEC* in *λProlog*. The resulting module, called `qcmec`, is displayed in Appendix A. The rule to diagnose hardware faults and an example from Section 3 are included in Appendices B and C. This code has been tested using the *Terzo* implementation of *λProlog*, version 1.0b, which is available from `http://www.cse.psu.edu/~dale/lProlog/`.

We define a family of representation functions $\ulcorner\cdot\urcorner$ that relate the mathematical entities we have been using in Section 2 to terms in *λProlog*. Specifically, we will need to encode *EC*-structures, the associated orderings, and the language of *QCMEC*. In the remainder of this section, we will refer to a generic *EC*-structure $\mathcal{H} = (E,\ P,\ [\cdot\rangle,\ \langle\cdot],\ ]\cdot,\cdot[)$.

We represent $\mathcal{H}$ by giving an encoding of the entities that constitute it. We introduce the types `event` and `property` so that every event in E (property in P) is represented by a distinct constant of type `event` (of type `property`). Event variables are represented as *λProlog* variables of the relative type. The initiation, termination and exclusivity relations, and event occurrences (traditionally represented in *EC*) are mapped to the predicate symbol `initiates`, `terminates`, `exclusive`, and `happens`, respectively, applied to the appropriate arguments. Declarations for these constants can be found in Appendix A.

For implementation purposes, it is more convenient to compute the relative ordering of two events on the basis of fragmented data (a binary acyclic relation) than to maintain this information as a strict order. We rely on the binary predicate symbol `beforeFact` to represent the edges of the binary acyclic relation. We encapsule the clauses for the predicate `before`, which implements its transitive closure, in the module `transClo`. We do not show details for space reasons, but a quadratic implementation can be found in [2].

In order to encode the syntax of *QCMEC*, we define the type `mvi`, intended to represent the formulas of this language (as opposed to the formulas of *λProlog*, that have type `o`). The representation of formulas is then relatively standard [4], except for quantifiers:

$$
\begin{aligned}
\ulcorner \bar{p}(\bar{e}_1, \bar{e}_2) \urcorner &= \texttt{period}\, \ulcorner \bar{e}_1 \urcorner\, \ulcorner \bar{p} \urcorner\, \ulcorner \bar{e}_2 \urcorner \\
\ulcorner \bar{e}_1 < \bar{e}_2 \urcorner &= \ulcorner \bar{e}_1 \urcorner\, \texttt{precedes}\, \ulcorner \bar{e}_2 \urcorner \\
\ulcorner \neg\varphi \urcorner &= \texttt{neg}\, \ulcorner \varphi \urcorner \\
\ulcorner \varphi_1 \wedge \varphi_2 \urcorner &= \ulcorner \varphi_1 \urcorner\, \texttt{and}\, \ulcorner \varphi_2 \urcorner \\
\ulcorner \varphi_1 \vee \varphi_2 \urcorner &= \ulcorner \varphi_1 \urcorner\, \texttt{or}\, \ulcorner \varphi_2 \urcorner \\
\ulcorner \varphi_1 \supset \varphi_2 \urcorner &= \ulcorner \varphi_1 \urcorner\, \texttt{implies}\, \ulcorner \varphi_2 \urcorner \\
\ulcorner \Box\varphi \urcorner &= \texttt{must}\, \ulcorner \varphi \urcorner \\
\ulcorner \Diamond\varphi \urcorner &= \texttt{may}\, \ulcorner \varphi \urcorner \\
\ulcorner \forall x.\varphi \urcorner &= \texttt{forAllEvent}\ (x \setminus \ulcorner \varphi \urcorner) \\
\ulcorner \exists x.\varphi \urcorner &= \texttt{forSomeEvent}\ (x \setminus \ulcorner \varphi \urcorner)
\end{aligned}
$$

Quantifiers differ from the other syntactic entities of a language such as *QCMEC* by the fact that they *bind* a variable in their argument (e.g. x in $\exists x.\varphi$). Bound variables are then subject to implicit renaming to avoid conflicts and to substitution. Encoding binding constructs in traditional programming languages such as *Prolog* is painful since these operations must be explicitly programmed. *λProlog* and other higher-order languages permit a much leaner emulation since λ-abstraction $(x \setminus M)$ is itself a binder and their implementations come equipped with (efficient) ways of handling it. The idea, known as ***higher-order abstract syntax*** [10], is then to use *λProlog*'s abstraction mechanism as a universal binder. Binding constructs in the object language are then expressed as constants that take a λ-abstracted term as their argument (for example `forSomeEvent` is declared of type `(event -> mvi) -> mvi)`. Variable renaming happens behind the scene, and substitution is delegated to the meta-language as β-reduction.

An example will shed some light on this technique. Consider the formula $\varphi = \exists x.p(x, e_2)$, which representation is

```
forSomeEvent (x  (period x p e2))
```

where we have assumed that p and e_2 are encoded as the constants `p` and `e2` of the appropriate type. It is easy to convince oneself that this expression is well-typed. In order to ascertain the truth of φ, we need to check whether $p(e, e_2)$ holds for successive $e \in E$ until such an event is found. Automating this implies that, given a candidate event e_1 (represented as `e1`), we need to substitute `e1` for `x` in `period x p e2`. This can however be achieved by simply applying the argument of `forSomeEvent` to `e1`. Indeed, `(x \ (period x p e2)) e1` is equal to `period e1 p e2`, modulo β-reduction. This technique is used in clauses *12–13* in our implementation.

We represent the truth of a formula in *QCMEC* by means of the predicate `holds`. Clauses *1* to *13* in Appendix A implement the specification of this language given in Section 2. More precisely, clauses *1* and *2* provide a direct encoding of Definition 2.1, where clause *2* faithfully emulates the meta-predicate *br*. Clause *3* captures the meaning of the precedence construct, while clauses *4* to *7* reduce the truth check for the connectives of *QCMEC* to the derivability of the corresponding *λProlog* constructs. Notice that implication is translated back to a combination of negation and disjunction in clause *7*. Clauses *8* to *11* implement the semantics of the modalities as the recursive visit of all the extensions of the current knowledge state; further details can be found in [4]. Existential quantifiers are handled similarly to connectives in clause *12*. Although *λProlog* offers a form of universal quantification, we are forced to take a detour and express our universal quantifiers as negations and existentials in clause *13*. A lengthy discussion of the logical reasons behind this step can be found in [4].

4.3 Soundness and Completeness

The encoding we have chosen as an implementation of *QCMEC* permits an easy proof of its faithfulness with respect to the formal specification of this formalism. Key factors in the feasibility of this endeavor are the precise semantic definition of *QCMEC* given in Section 2, and the exploitation of the declarative features of *λProlog*.

We only show the statement of our soundness and completeness result since a fully worked out proof would require a very detailed account of the semantics of *λProlog*, and is rather long, although simple. Space constraints prevent us from doing so. The interested reader can find the full development of a proof that relies on the same techniques in [4].

Theorem 4.1 (*Soundness and completeness of* qcmec)

Let $\mathcal{H} = (E,\ P,\ [\cdot\rangle,\ \langle\cdot],\ ]\cdot,\cdot[)$ be an EC-structure, o a binary acyclic relation over E and φ and formula in $\mathcal{L}_{\mathcal{H}}(QCMEC)$, then

$$\texttt{qcmec}, \ulcorner \mathcal{H} \urcorner, \ulcorner o \urcorner \vdash \texttt{holds}\, \varphi \quad \textit{iff} \quad \mathcal{I}_{\mathcal{H}}; o^+ \models \varphi. \quad \blacksquare$$

5 Complexity Analysis

This section is dedicated to studying the complexity of the various modal event calculi presented in Section 2. We assume the reader familiar with computational complexity theory [12]. Given an *EC*-structure

$\mathcal{H}$, a knowledge state $w \in W_{\mathcal{H}}$ and a formula φ relative to any of the modal event calculi presented in Section 2, we want to characterize the complexity of the problem of establishing whether $\mathcal{I}_{\mathcal{H}}; w \models \varphi$ is true, which is an instance of the general problem of model checking.

We model our analysis around the truth relations given in Definitions 2.2, 2.3 and 2.5. We measure the complexity of testing whether $\mathcal{I}_{\mathcal{H}}; w \models \varphi$ holds in terms of the size n of the input structure (where n is the number of recorded events) and the size k of the input formula (without loss of generality, we sometimes interpret k as the number of atomic formulas occurring in φ).

The notion of cost we adopt is as follows: we assume that verifying the truth of the propositions $e \in [p\rangle$ and $e \in \langle p]$ costs $\mathcal{O}(1)$. Although possible in principle, it is disadvantageous to maintain knowledge states as transitive relations. We instead record an acyclic binary relation o on events whose transitive closure o^+ is w. Verifying whether $e_1 <_w e_2$ holds becomes a reachability problem in o and it can be solved in quadratic time $O(n^2)$ in the number n of events [2]. The cost of solving the query $e_1 < e_2$ is therefore quadratic.

We begin our analysis from the plain Event Calculus. Model checking in *EC* (Definition 2.2) is a polynomial task and costs $\mathcal{O}(n^3)$ [2, 5].

Theorem 5.1 (*Cost of model checking in EC*)

Model checking in EC has complexity $\mathcal{O}(n^3)$. ■

We obtain the same bound if we allow property-labeled intervals $p(e_1, e_2)$, possibly prefixed with at most one modal operator [5]. This bound does not change if we consider precedence queries.

An *ECMEC*-formula is the boolean combination of a number of atomic formulas, i.e. property-labeled intervals $p(e_1, e_2)$ or precedence tests $e_1 < e_2$, possibly prefixed with a modal operator.

Given an *ECMEC*-formula that contains k atomic formulas, checking it reduces to testing k atomic formulas, possibly prefixed with a modal operator, each of them is solved in $\mathcal{O}(n^3)$. Thus, model checking in *ECMEC* has polynomial complexity.

Theorem 5.2 (*Cost of model checking in ECMEC*)

Model checking in ECMEC has complexity $\mathcal{O}(kn^3)$. ■

Model checking in *CMEC* (Definition 2.3) involves an exhaustive exploration of the extensions of the current knowledge state, whose number is, in general, exponential in the number of events. This raises complexity beyond tractability.

Theorem 5.3 (*Cost of model checking in CMEC*)

Model checking in CMEC is **PSPACE**-*complete.*

Proof. In order to prove that model checking in *CMEC* is in **PSPACE**, we show that this problem belongs to **AP**, that is, we define an alternating polynomial time algorithm that solves it.

Let φ be a *CMEC*-formula and w a knowledge state. If $\varphi = \alpha \wedge \beta$ (resp. $\varphi = \alpha \vee \beta$), the algorithm enters in an AND (resp. OR) state. It nondeterministically chooses one among α and β and evaluates it in w. If $\varphi = \neg(\alpha \wedge \beta)$ (resp. $\varphi = \neg(\alpha \vee \beta)$), the algorithm evaluates $\neg\alpha \vee \neg\beta$ (resp. $\neg\alpha \wedge \neg\beta$). If $\varphi = \neg\neg\alpha$, the algorithm verifies α. If $\varphi = \Box\alpha$ (resp. $\varphi = \Diamond\alpha$), the algorithm enters in an AND (resp. OR) state. It determines all the extensions of w and it nondeterministically chooses one, say w'. Then, it evaluates α in w'. If $\varphi = \neg\Box\alpha$ (resp. $\varphi = \neg\Diamond\alpha$), the algorithm evaluates $\Diamond\neg\alpha$ (resp. $\Box\neg\alpha$). If $\varphi = p(e_1, e_2)$ (resp. $\varphi = \neg p(e_1, e_2)$), the algorithm accepts it if and only if all points (resp. at least one point) from *i* to *iv* of Definition 2.2 hold (resp. does not hold). Finally, if $\varphi = e_1 < e_2$ (resp. $\varphi = \neg(e_1 < e_2)$), the algorithm accepts if and only if $e_1 <_w e_2$ (resp. $e_1 \not<_w e_2$).

It follows, from the definition of acceptance of alternating machines [12], that a *CMEC*-instance $(\mathcal{H}, \varphi, w)$ is accepted if and only if $I_{\mathcal{H}}; w \models \varphi$. Moreover, the time needed is polynomial in the size of $\mathcal{H}$ and φ. Thus, model checking in *CMEC* in **AP**. Since **AP** = **PSPACE** [12], it is in **PSPACE**.

In order to prove that the considered problem is **PSPACE**-hard, we define a (polynomial) reduction of QSAT [12] into *CMEC*.

Let $G = \exists x_1.\forall x_2.\exists x_3.\forall x_4.\ldots Qx_n.F(x_1, x_2, \ldots x_n)$, with $n \geq 1$, be a quantified Boolean formula where the quantifiers alternate, so that Q is $\exists$ ($\forall$) if n is odd (even).

We now define the *EC*-structure $\mathcal{H} = (E, P, [\cdot\rangle, \langle\cdot],]\cdot, \cdot[)$ such that:

- $E = \{e_{x_i}, e_{\neg x_i} : 1 \leq i \leq n\}$;
- $P = \{p_{x_i} : 1 \leq i \leq n\}$;
- $[p_{x_i}\rangle = \{e_{x_i}\}$, $\langle p_{x_i}] = \{e_{\neg x_i}\}$, for $1 \leq i \leq n$;
- $]\cdot, \cdot[= \emptyset$.

Moreover, let $w = \emptyset$, and $\hat{F}$ be the formula obtained replacing in $F(x_1, x_2, \ldots x_n)$ every occurrence of a variable x_i with $p_{x_i}(e_{x_i}, e_{\neg x_i})$.

Further, consider the following recursive definition of the *CMEC*-formula F_k:

$$F_k = \begin{cases} \Diamond\hat{F} & k = n \\ \Diamond\Box\hat{F} & k = n-1 \\ \Diamond\Box(\psi_{k+2} \supset F_{k+2}) & 1 \leq k < n-1 \end{cases}$$

where

$$\psi_k = \bigwedge_{k \leq i \leq n} \neg(e_{x_i} < e_{\neg x_i}) \wedge \neg(e_{\neg x_i} < e_{x_i}).$$

Observe that, if $w \models \psi_k$, then, for every i from k to n, the events e_{x_i} and $e_{\neg x_i}$ are unordered. It is possible to prove that, for $\varphi = F_1$, $w \models \varphi$ if and only if G is true. ■

In the following, we analyze the complexity of the quantified calculi defined in Section 2. We begin our analysis with the complexity of *EQCMEC*, i.e. the quantified version of *ECMEC*. We have proved that model checking in *ECMEC* is polynomial time-bounded (Theorem 5.2). However, the extension of *ECMEC* with quantifiers arises complexity beyond **P**. In particular, model checking in *EQCMEC* is **PSPACE**-complete, as proved by the following theorem.

Theorem 5.4 (*Cost of model checking in EQCMEC*)

Model checking in EQCMEC is **PSPACE**-*complete.*

Proof. In order to prove that model checking in *EQCMEC* is in **PSPACE**, we show that it belongs to **AP**. In order to do so, we extend the alternating polynomial time algorithm used in the proof of Theorem 5.3. If $\varphi = \forall x.\alpha$ (resp. $\varphi = \exists x.\alpha$), the algorithm enters in an AND (resp. OR) state. It nondeterministically chooses one event, say e, and evaluates the formula obtained by replacing all occurrences of x in α that are in the scope of the quantifier by e. If $\varphi = \neg\forall x.\alpha$ (resp. $\varphi = \neg\exists x.\alpha$), the algorithm evaluates $\exists x.\neg\alpha$ (resp. $\forall x.\neg\alpha$).

From the definition of acceptance of alternating machines [12], it follows that an *EQCMEC*-instance $(\mathcal{H}, \varphi, w)$ is accepted if and only if $I_{\mathcal{H}}; w \models \varphi$. Moreover, the time needed is polynomial in the size of $\mathcal{H}$ and φ. Thus, model checking in *EQCMEC* is in **AP**. Since **AP** = **PSPACE** [12], it is in **PSPACE**.

In order to prove that the considered problem is **PSPACE**-hard, we define a (polynomial) reduction of QSAT [12] into *EQCMEC*.

Let $G = \exists x_1.\forall x_2.\exists x_3.\forall x_4.\ldots Qx_n.F(x_1, x_2, \ldots x_n)$, with $n \geq 1$, be a quantified boolean formula where the quantifiers alternate (so that Q is $\exists$ ($\forall$) if n is odd (even)).

We then define an *EC*-structure $\mathcal{H} = (E, P, [\cdot\rangle, \langle\cdot],]\cdot,\cdot[)$ such that:

- $E = \{e_1, e_2, \ldots e_n, e\}$;
- $P = \{p_1, p_2, \ldots p_n\}$;
- $[p_i\rangle = \{e_i\}$ and $\langle p_i] = \{e\}$, for $1 \leq i \leq n$;
- $]\cdot,\cdot[= \emptyset$.

Moreover, let $w = \{(e_i, e) : 1 \leq i \leq n\}$, and

$$\varphi = \exists x_1.\forall x_2.\exists x_3.\forall x_4.\ldots Q_n x_n.\hat{F}(x_1, x_2, \ldots x_n)$$

where $\hat{F}(x_1, x_2, \ldots x_n)$ is obtained replacing every occurrence of a variable x_i in the formula $F(x_1, x_2, \ldots x_n)$ with $p_i(x_i, e)$. Notice that φ is an *EQCMEC*-formula. In particular, it has no modal operator. It is not difficult to see that $w \models \varphi$ if and only if G is true. ■

In the following, we characterize the explicit time complexity of model checking in *EQCMEC*. Since model checking in *EQCMEC* is **PSPACE**-complete, we expect an exponential bound.

We will exploit the unfolding lemma (2.6). This result affirms that every formula involving one event quantifier at its top-level can be replaced with the conjunction or disjunction of n instances of it, where n is the number of events. If we have a nesting of q such quantifiers, we are led to solve n^q instances. In general, if we eliminate in this manner all event quantifiers in a formula φ of size k, we will produce a formula φ' without quantifiers, i.e. an *ECMEC* formula, of size at most kn^k. Taking advantage of Theorem 5.2, we get the following upper bound.

Theorem 5.5 (*Upper bound for time complexity of model checking in EQCMEC*)

Model checking in EQCMEC has complexity $\mathcal{O}(kn^{k+3})$. ■

Practical applications using modal event calculi with quantifiers are expected to model situations involving a large number of events, while the size of the queries will in general be limited. The hardware fault diagnosis example in Section 3 falls into this category. In such contexts, the fact that *EQCMEC* is polynomial in the number of events is essential. At worst, the dependence of the exponent on k may lead to polynomials of high degree.

Finally, we consider the calculus *QCMEC*. Since *EQCMEC* is a linguistic fragment of *QCMEC*, model checking in *QCMEC* is **PSPACE**-hard. Nevertheless, it is possible to show that model checking in *QCMEC* is polynomial space-bounded.

Theorem 5.6 (*Cost of model checking in QCMEC*)

Model checking in QCMEC is **PSPACE**-*complete.*

Proof. To see that the considered problem is in **PSPACE**, we exploit the polynomial alternating algorithm defined in the proof of Theorem 5.4 for *EQCMEC*. It works correctly for *QCMEC* as well.

Since *EQCMEC* is a linguistic fragment of *QCMEC* and *EQCMEC* is **PSPACE**-hard (Theorem 5.4), model checking in *QCMEC* is **PSPACE**-hard. ■

In Section 4 we have transliterated the definition of *QCMEC* and its subcalculi in the higher-order logic programming language *λProlog* [10]. The directness of the implementation allows checking easily that the complexity of the implemented algorithms coincide with the bounds proved in this section for the problems they implement.

6 Conclusions and Future Work

In this paper, we have extended a number of modal event calculi [1, 5, 9] with the possibility of using quantifiers and precedence tests in queries. The net effect of these combined additions has been a substantial gain in expressiveness. The extra computational cost was shown acceptable for queries of a reasonable size in those subcalculi that are tractable without quantifiers. We have implemented the resulting formalisms in the higher-order logic programming language *λProlog* [10], which we used to encode case studies from the area of hardware and medical diagnosis.

We intend gaining a better understanding of the interactions among the various operators of our calculi, in particular between quantifiers and modalities, in order to devise simplifications of costly queries and thus better implementations. We also intend studying the integration of preconditions [6].

Acknowledgments

The first author was supported by ONR grant N00014-97-1-0505, Multidisciplinary University Research Initiative *Semantic Consistency in Information Exchange*. The work of the third author was partially supported by the CNR project *Programmazione logica: strumenti per analisi e trasformazione di programmi; tecniche di ingegneria del software; estensioni con vincoli, concorrenza ed oggetti (STE).*

References

[1] Iliano Cervesato, Luca Chittaro, and Angelo Montanari. A modal calculus of partially ordered events in a logic programming framework. In L. Sterling, editor, *Proceedings of the Twelfth International Conference on Logic Programming — ICLP'95*, pages 299–313, Kanagawa, Japan, 13–16 June 1995. MIT Press.

[2] Iliano Cervesato, Luca Chittaro, and Angelo Montanari. Speeding up temporal reasoning by exploiting the notion of kernel of an ordering relation. In S.D. Goodwin and H.J. Hamilton, editors, *Proceedings of the Second International Workshop on Temporal Representation and Reasoning — TIME'95*, pages 73–80, Melbourne Beach, FL, 26 April 1995.

[3] Iliano Cervesato, Luca Chittaro, and Angelo Montanari. A general modal framework for the event calculus and its skeptical and credulous variants. In W. Wahlster, editor, *Proceedings of the Twelfth European Conference on Artificial Intelligence — ECAI'96*, pages 33–37, Budapest, Hungary, 12–16 August 1996. John Wiley & Sons.

[4] Iliano Cervesato, Luca Chittaro, and Angelo Montanari. A general modal framework for the event calculus and its skeptical and credulous variants. Technical Report 37/96-RR, Dipartimento di Matematica e Informatica, Università di Udine, July 1996. Submitted for publication.

[5] Iliano Cervesato, Massimo Franceschet, and Angelo Montanari. A hierarchy of modal event calculi: Expressiveness and complexity. In H. Barringer, M. Fisher, D. Gabbay, , and G. Gough, editors, *Proceedings of the Second International Conference on Temporal Logic — ICTL'97*, pages 1–17, Manchester, England, 14–18 July 1997. Kluwer, Applied Logic Series. To appear.

[6] Iliano Cervesato, Massimo Franceschet, and Angelo Montanari. Modal event calculi with preconditions. In R. Morris and L. Khatib, editors, *Fourth International Workshop on Temporal Representation and Reasoning — TIME'97*, pages 38–45, Daytona Beach, FL, 10–11 May 1997. IEEE Computer Society Press.

[7] Iliano Cervesato, Angelo Montanari, and Alessandro Provetti. On the non-monotonic behavior of the event calculus for deriving maximal time intervals. *International Journal on Interval Computations*, 2:83–119, 1993.

[8] Luca Chittaro, Angelo Montanari, and Alessandro Provetti. Skeptical and credulous event calculi for supporting modal queries. In A. Cohn, editor, *Proceedings of the Eleventh European Conference on Artificial Intelligence — ECAI'94*, pages 361–365. John Wiley & Sons, 1994.

[9] Robert Kowalski and Marek Sergot. A logic-based calculus of events. *New Generation Computing*, 4:67–95, 1986.

[10] Dale Miller. Lambda Prolog: An introduction to the language and its logic. Current draft available from `http://cse.psu.edu/~dale/lProlog`, 1996.

[11] K. Nökel. *Temporarily Distributed Symptoms in Technical Diagnosis*. Springer-Verlag, 1991.

[12] Christos Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[13] Leon Sterling and Ehud Shapiro. *The Art of Prolog: Advanced Programming Techniques*. MIT Press, 1994.

A Implementation of *QCMEC*

```
module qcmec.
accumulate transClo.

kind event    type.
kind property type.
kind mvi      type.

type initiates  event -> property -> o.
type terminates event -> property -> o.
type exclusive  property -> property -> o.
type happens    event -> o.

% ------- MVIs
type period  event -> property -> event -> mvi.
type holds   mvi -> o.
type broken  event -> property -> event -> o.

holds (period Ei P Et) :-                              % 1 %
        happens Ei, initiates Ei P,
        happens Et, terminates Et P,
        before Ei Et, not (broken Ei P Et).
broken Ei P Et :-                                      % 2 %
        happens E,
        before Ei E, before E Et,
        (initiates E Q; terminates E Q),
        (exclusive P Q; P = Q).

% ------- Ordering
type precedes  event -> event -> mvi.  infixr precedes 6.

holds (E1 precedes E2) :-  before E1 E2.               % 3 %

% ------- Connectives
type neg      mvi -> mvi.
type and      mvi -> mvi -> mvi.      infixr and     5.
type or       mvi -> mvi -> mvi.      infixr or      5.
type implies  mvi -> mvi -> mvi.      infixl implies 4.

holds (neg X) :-  not (holds X).                       % 4 %
holds (X and Y) :-  holds X, holds Y.                  % 5 %
holds (X or Y) :-  holds X; holds Y.                   % 6 %
holds (X implies Y) :-  holds ((neg X) or Y).          % 7 %
```

```
% ------- Modalities
type must        mvi -> mvi.
type may         mvi -> mvi.
type fails_must  mvi -> o.

holds (must X) :-  holds X, not (fails_must X).             % 8 %

fails_must X :-                                             % 9 %
        happens E1, happens E2, not (E1 = E2),
        not (before E1 E2), not (before E2 E1),
        beforefact E1 E2 => not (holds (must X)).

holds (may X) :-  holds X.                                 % 10 %
holds (may X) :-                                           % 11 %
        happens E1, happens E2, not (E1 = E2),
        not (before E1 E2), not (before E2 E1),
        beforefact E1 E2 => holds (may X).

% ------- Quantifiers
type forAllEvent  (event -> mvi) -> mvi.
type forSomeEvent (event -> mvi) -> mvi.

holds (forAllEvent X) :-                                   % 12 %
        not (sigma E \ (happens E, not (holds (X E)))).
holds (forSomeEvent X) :-  sigma E \ holds (X E).          % 13 %
```

B Diagnosing Hardware Faults

```
module cncc.
accumulate qcmec.

type c  property.
type r1 property.
type r2 property.
type phi1 o.
type phi2 o.

phi1 :- holds (forAllEvent E0 \
                 forAllEvent E5 \
                   ((period E0 c E5) implies
                      (forSomeEvent E1 \
                       forSomeEvent E2 \
                       forSomeEvent E3 \
                       forSomeEvent E4 \
                         (must ((E0 precedes E1)  and
                                (period E1 r1 E2) and
                                (E2 precedes E3)  and
                                (period E3 r2 E4) and
                                (E4 precedes E5)))))).

phi2 :- holds (forAllEvent E0 \
                 forAllEvent E5 \
                   ((period E0 c E5) implies
                      (forSomeEvent E1 \
                       forSomeEvent E2 \
                       forSomeEvent E3 \
                       forSomeEvent E4 \
                         (may ((E0 precedes E1)  and
                               (period E1 r1 E2) and
                               (E2 precedes E3)  and
                               (period E3 r2 E4) and
                               (E4 precedes E5)))))).
```

C A Specific Situation

```
module example.
accumulate cncc.

type e0 event.   happens e0.   initiates  e0 c.
type e1 event.   happens e1.   initiates  e1 r1.
type e2 event.   happens e2.   terminates e2 r1.
type e3 event.   happens e3.   initiates  e3 r2.
type e4 event.   happens e4.   terminates e4 r2.
type e5 event.   happens e5.   terminates e5 c.

beforefact e0 e1.   beforefact e1 e2.   beforefact e1 e3.
beforefact e2 e4.   beforefact e3 e4.   beforefact e4 e5.
```

Probabilistic Deduction with Conditional Constraints over Basic Events

Thomas Lukasiewicz
Institut für Informatik, Universität Gießen
Arndtstr. 2, D-35392 Gießen, Germany
E-mail: lukasiewicz@informatik.uni-giessen.de

Abstract

We show that probabilistic deduction with conditional constraints over basic events is NP-hard. We then focus on the special case of probabilistic deduction in conditional constraint trees. We elaborate very efficient and globally complete techniques for probabilistic deduction. More precisely, for exact conditional constraint trees, we present a local approach to globally complete probabilistic deduction, which runs in linear time in the size of the conditional constraint trees. For general conditional constraint trees, we show that globally complete probabilistic deduction can be done by solving global nonlinear programs. We elaborate how these nonlinear programs can be transformed into equivalent linear programs, which are solvable in polynomial time in the size of the conditional constraint trees.

1 INTRODUCTION

Representing and reasoning with uncertain knowledge has gained growing importance in the recent decades. The literature contains many different formalisms and methodologies for tackling uncertainty. Most of them are directly or indirectly based on probability theory.

In this paper, we focus on probabilistic deduction with conditional constraints (that is, interval restrictions for conditional probabilities). The considered probabilistic deduction problems consist of a probabilistic knowledge-base and a probabilistic query. We give a classical example. As a probabilistic knowledge-base, we may take the probabilistic knowledge that all ostriches are birds, that ostriches do not fly, that at least 95 per cent of all birds fly, and that not more than 10 per cent of all birds are ostriches. As a probabilistic query, we may wonder about the entailed greatest lower bound and the entailed least upper bound for the rate of all birds that are ostriches. The solution to this probabilistic deduction problem is 0 per cent for the entailed greatest lower bound, and 5 per cent for the entailed least upper bound.

There is a long controversy in the probabilistic community about whether to solve this kind of probabilistic deduction problems in a global approach by linear programming or in a local approach by the iterative application of inference rules. The global approach by linear programming (see, for example, [23], [13], [10], [17], [16], [3], [22], and [20]) can be performed within rich probabilistic languages capable of representing many facets of probabilistic knowledge (see especially [10]). Crucially, probabilistic deduction by linear programming is globally complete, that is, it really produces the requested tightest bounds entailed by the whole probabilistic knowledge-base. However, it generally runs in exponential time in the size of the probabilistic deduction problems. Furthermore, it cannot provide any explanatory informations on how the deduced results are obtained.

Mainly to overcome these deficiencies, researchers started to work on local techniques based on inference rules. The local approach (see, for example, [7], [9], [2], [8], [28], [11], [15], and [18]) is generally performed within more restricted probabilistic languages. The iterative application of inference rules is very rarely and only within very restricted languages globally complete (see [11] for an example of globally complete local probabilistic deduction in a very restricted framework). Whenever the languages become more expressive, the global completeness of local probabilistic deduction cannot be guaranteed anymore. Local techniques are generally expected to be more efficient than global ones. Moreover, they can elucidate the deduction process by the sequence of applied inference rules.

Probabilistic deduction with conditional constraints over ***propositional events*** is immediately NP-hard, since it is a generalization of the satisfiability problem for probabilistic logic [23], which is known to be NP-complete from [13]. As a first crucial contribution of this paper, we even show that probabilistic deduction with conditional constraints over *basic events* (like in the introductory example) is also NP-hard. It is surprising that this quite restricted class of probabilistic deduction problems is still computationally so difficult.

Hence, it is unlikely that there is an algorithm that efficiently solves all probabilistic deduction problems with conditional constraints over basic events. However, we can still hope that there are efficient special-case, average-case, or approximation algorithms.

In this paper, we focus on elaborating an efficient special-case algorithm. We concentrate on probabilistic deduction in conditional constraint trees. It is an interesting subclass of all probabilistic deduction problems with conditional constraints over basic events. Conditional constraint trees are undirected trees with basic events as nodes and with bidirectional conditional constraints over basic events as edges between the nodes. Like Bayesian networks, conditional constraint trees represent a well-structured probabilistic knowledge-base. However, differently from Bayesian networks, they do not require any knowledge about probabilistic independence.

For exact conditional constraint trees, we present functions for deducing tightest lower and upper bounds in linear time in the size of the conditional constraint trees. For general conditional constraint trees, we show that greatest lower bounds can be deduced in the same way, in linear time in the size of the conditional constraint trees. However, computing least upper bounds turns out to be computationally more difficult. It can be done by solving special nonlinear programs. We show how these nonlinear programs can be transformed into equivalent linear programs. Crucially, these linear programs have a number of variables and of linear inequalities being linear and polynomial, respectively, in the size of the conditional constraint trees. Thus, our way of deducing least upper bounds still runs in polynomial time in the size of the conditional constraint trees, since linear programming runs in polynomial time in the size of the linear programs.

Another contribution of this paper is related to the controversy about whether to perform probabilistic deduction with conditional constraints by linear programming or by the iterative application of inference rules. On the one hand, we provide a very efficient technique for globally complete probabilistic deduction in exact conditional constraint trees. However, on the other hand, we also show that extending this local technique to general conditional constraint trees already results in solving global linear programs.

Previous work on similar deduction problems is rare in the literature, the few existing does not provide any or just very few results towards global completeness. In [2], the rule of quantified syllogism and the generalized Bayes' rule are applied to sets of bidirectional conditional constraints over basic events. This deduction technique is generally not globally complete. In [27], trees of bidirectional conditional constraints over basic events are examined within the framework of deductive database systems. A system of inference rules is presented, which is globally complete only in deducing lower bounds.

The rest of this paper is organized as follows. In Section 2, we formulate the probabilistic deduction problems considered in this work. Section 3 focuses on complete probabilistic deduction in exact and general conditional constraint trees. In Section 4, we summarize the main results of this work.

This paper is a revised extract of own work from [19], which we extended by new results on the computational complexity of probabilistic deduction with conditional constraints over basic events. Moreover, we improved significantly the probabilistic deduction technique for general conditional constraint trees.

2 FORMULATING THE DEDUCTION PROBLEMS

In this section, we introduce the syntactic and semantic notions related to probabilistic knowledge in general and to conditional constraint trees in particular.

2.1 PROBABILISTIC KNOWLEDGE

Before focusing on the details of conditional constraint trees, we give a general introduction to the kind of probabilistic knowledge considered in this work. We deal with conditional constraints over propositional events. They represent interval restrictions for conditional probabilities of propositional events. The formal background introduced in this section is commonly accepted in the literature (see, for example, [11] for other work in the same spirit).

We assume a nonempty and finite set of ***basic events*** $\mathcal{B} = \{B_1, B_2, \ldots, B_n\}$. The set of ***conjunctive events*** $\mathcal{C}_{\mathcal{B}}$ is the closure of $\mathcal{B}$ under the Boolean operation $\wedge$. We abbreviate the conjunctive event $C \wedge D$ by CD. The set of ***propositional events*** $\mathcal{G}_{\mathcal{B}}$ is the closure of $\mathcal{B}$

under the Boolean operations $\wedge$ and $\neg$. We abbreviate the propositional events $G \wedge H$ and $\neg G$ by GH and $\overline{G}$, respectively. The *true event* $B_1 \vee \neg B_1$ and the *false event* $B_1 \wedge \neg B_1$ are abbreviated by $\top$ and $\bot$, respectively. *Conditional constraints* are expressions of the form $(H|G)[u_1, u_2]$ with real numbers $u_1, u_2 \in [0,1]$ and propositional events G and H. In $(H|G)[u_1, u_2]$, we call G the *premise* and H the *conclusion*.

To define probabilistic interpretations of propositional events and of conditional constraints, we introduce atomic events and the binary relation $\Rightarrow$ between atomic and propositional events. The set of *atomic events* $\mathcal{A}_\mathcal{B}$ is defined by $\mathcal{A}_\mathcal{B} = \{E_1 E_2 \cdots E_n \,|\, E_i = B_i$ or $E_i = \overline{B}_i$ for all $i \in [1\!:\!n]\}$. The atomic events of our framework coincide with the more commonly known possible worlds from probabilistic logic [23]. For all atomic events A and all propositional events G, let $A \Rightarrow G$ iff $A\overline{G}$ is a propositional contradiction.

A *probabilistic interpretation* Pr is a mapping from $\mathcal{A}_\mathcal{B}$ to $[0,1]$ such that all $Pr(A)$ with $A \in \mathcal{A}_\mathcal{B}$ sum up to 1. Pr is extended in a well-defined way to propositional events G by $Pr(G) = \sum_{A \in \mathcal{A}_\mathcal{B},\, A \Rightarrow G} Pr(A)$. Pr is extended to conditional constraints by:

$$Pr \models (H|G)[u_1, u_2] \text{ iff} \\ u_1 \cdot Pr(G) \leq Pr(GH) \leq u_2 \cdot Pr(G)\,.$$

Note that $Pr(G) = 0$ entails $Pr \models (H|G)[u_1, u_2]$ (see, for example, [14] and [11] for other work with the same semantics of conditional probability statements).

The notions of models, satisfiability, and logical consequence for conditional constraints are defined in the classical way. A probabilistic interpretation Pr is a *model* of a conditional constraint $(H|G)[u_1, u_2]$ iff $Pr \models (H|G)[u_1, u_2]$. Pr is a *model* of a set of conditional constraints KB, denoted $Pr \models KB$, iff Pr is a model of all $(H|G)[u_1, u_2] \in KB$. KB is *satisfiable* iff a model of KB exists. $(H|G)[u_1, u_2]$ is a *logical consequence* of KB, denoted $KB \models (H|G)[u_1, u_2]$, iff each model of KB is also a model of $(H|G)[u_1, u_2]$.

For a conditional constraint $(H|G)[u_1, u_2]$ and a set of conditional constraints KB, let $\boldsymbol{u}$ denote the set of all real numbers $u \in [0,1]$ for which there exists a model Pr of KB with $u \cdot Pr(G) = Pr(GH)$ and $Pr(G) > 0$. Now, we easily verify that $(H|G)[u_1, u_2]$ is a logical consequence of KB iff $u_1 \leq \inf \boldsymbol{u}$ and $u_2 \geq \sup \boldsymbol{u}$.

This observation yields a canonic notion of tightness for logical consequences of conditional constraints. The conditional constraint $(H|G)[u_1, u_2]$ is a *tight logical consequence* of KB, denoted $KB \models_{tight} (H|G)[u_1, u_2]$, iff $u_1 = \inf \boldsymbol{u}$ and $u_2 = \sup \boldsymbol{u}$.

The set $\boldsymbol{u}$ is a closed interval in the real numbers (see, for example, [11] and [19]). Note that for $\boldsymbol{u} = \emptyset$, we assume $\inf \boldsymbol{u} = 1$ and $\sup \boldsymbol{u} = 0$. Thus, we get $\boldsymbol{u} = \emptyset$ iff $KB \models (G|\top)[0,0]$ iff $KB \models_{tight} (H|G)[1,0]$ iff $KB \models (H|G)[u_1, u_2]$ for all $u_1, u_2 \in [0,1]$.

Based on the just introduced notion of tightness, probabilistic deduction problems and their solutions are more formally specified as follows.

A *probabilistic knowledge-base* $(\mathcal{B}, KB)$ consists of a set of basic events $\mathcal{B}$, and a set of conditional constraints KB over $\mathcal{G}_\mathcal{B}$ with $u_1 \leq u_2$ for all $(H|G)[u_1, u_2] \in KB$. A *probabilistic query* to a probabilistic knowledge-base $(\mathcal{B}, KB)$ is an expression of the form $\exists(F|E)[x_1, x_2]$ with $E, F \in \mathcal{G}_\mathcal{B}$ and two different variables x_1 and x_2. Its *tight answer* is the substitution $\sigma = \{x_1/u_1,\, x_2/u_2\}$ with $u_1, u_2 \in [0,1]$ such that $KB \models_{tight} (F|E)[u_1, u_2]$. A *correct answer* is a substitution $\sigma = \{x_1/u_1,\, x_2/u_2\}$ with $u_1, u_2 \in [0,1]$ such that $KB \models (F|E)[u_1, u_2]$.

Finally, we define the notions of soundness and of completeness related to inference rules and to techniques for probabilistic deduction. An inference rule $KB \vdash (H|G)[u_1, u_2]$ is *sound* iff $KB \models (H|G)[u_1, u_2]$, where $(H|G)[u_1, u_2]$ is a conditional constraint and KB is a set of conditional constraints. It is sound and *locally complete* iff $KB \models_{tight} (H|G)[u_1, u_2]$. A technique for probabilistic deduction is *sound* iff it computes a correct answer to any given probabilistic query. It is sound and *globally complete* iff it computes the tight answer to any given probabilistic query.

2.2 COMPUTATIONAL COMPLEXITY

In the framework of conditional constraints over propositional events, the problem of computing the tight answer to a probabilistic query is NP-hard, since it generalizes the satisfiability problem for probabilistic logic, which is known to be NP-complete from [13].

Surprisingly, the problem of computing the tight answer to a probabilistic query remains NP-hard even if we just use conditional constraints over basic events.

Theorem 2.1 *The problem of computing the tight answer to a probabilistic query over basic events that is directed to a probabilistic knowledge-base over basic events is NP-hard.*

Proof. The NP-complete problem of graph 3-colorability [12] can be polynomially-reduced to the problem of computing the tight answer to a probabilistic query over basic events that is directed to a probabilistic knowledge-base over basic events. The proof follows similar lines to the proof of NP-hardness of 2PSAT given in [13].

Let (V, E) be an undirected graph. We construct a probabilistic knowledge-base $(\mathcal{B}, KB)$ as follows. We initialize $(\mathcal{B}, KB)$ with $(\{B\}, \emptyset)$. For each node $v \in V$, we increase $\mathcal{B}$ by the new basic events B_v^1, B_v^2, and B_v^3. For each node $v \in V$ and for each $i \in \{1,2,3\}$, we increase KB by $(B|B_v^i)[1,1]$ and $(B_v^i|B)[1/3,1/3]$. For each node $v \in V$ and for each $i, j \in \{1,2,3\}$ with $i < j$, we increase KB by $(B_v^j|B_v^i)[0,0]$. For each edge $\{u,v\} \in E$ and for each $i \in \{1,2,3\}$, we increase KB by $(B_v^i|B_u^i)[0,0]$. Now, we show that (V,E) is 3-colorable iff $\{x_1/1,\ x_2/1\}$ is the tight answer to the probabilistic query $\exists(B|B)[x_1, x_2]$ to $(\mathcal{B}, KB)$, or equivalently, iff KB is satisfiable:

If (V,E) is 3-colorable, then there exists a mapping c_1 from V to $\{1,2,3\}$ with $c_1(u) \neq c_1(v)$ for all edges $\{u,v\} \in E$. Thus, if π is a cyclic permutation of the members in $\{1,2,3\}$ and if $c_2, c_3 \colon V \to \{1,2,3\}$ are defined by $c_2(v) = \pi(c_1(v))$ and $c_3(v) = \pi(c_2(v))$ for all nodes $v \in V$, then also $c_2(u) \neq c_2(v)$ and $c_3(u) \neq c_3(v)$ for all edges $\{u,v\} \in E$. For $j \in \{1,2,3\}$, let $A_j \in \mathcal{A}_\mathcal{B}$ such that $A_j \Rightarrow B$ and $A_j \Rightarrow B_v^i$ iff $c_j(v) = i$ for all nodes $v \in V$ and $i \in \{1,2,3\}$. If $Pr\colon \mathcal{A}_\mathcal{B} \to [0,1]$ is defined by $Pr(A) = 1/3$ for all $A \in \{A_1, A_2, A_3\}$ and by $Pr(A) = 0$ for all $A \in \mathcal{A}_\mathcal{B} \setminus \{A_1, A_2, A_3\}$, then Pr is a model of KB. Conversely, if there is a model Pr of KB, then there is an atomic event $A \in \mathcal{A}_\mathcal{B}$ with $Pr(A) > 0$. Thus, if $c\colon V \to \{1,2,3\}$ is defined by $c(v) = i$ iff $A \Rightarrow B_v^i$ for all nodes $v \in V$, then $c(u) \neq c(v)$ for all edges $\{u,v\} \in E$. Hence, (V,E) is 3-colorable. □

Hence, it is unlikely that there is an efficient algorithm for computing the tight answer to all probabilistic queries over basic events that are directed to any given probabilistic knowledge-base over basic events. However, there may be still efficient algorithms for solving more specialized probabilistic deduction problems.

The rest of this work deals with probabilistic deduction in conditional constraint trees. The next section provides a motivating example, which gives evidence of the practical importance of this kind of probabilistic deduction problems.

2.3 MOTIVATING EXAMPLE

A senior student in mathematics describes his experience about being successful at the university as follows. The success of a student (su) is influenced by how well-informed (wi) and how well-prepared (wp) the student is. Well-informedness can be reached by interviewing professors (pr) or by asking senior students (st). Being well-prepared is influenced by how much time is invested in books (bo), exercises (ex), and hobbies (ho). It is estimated that between 60 and 70 per cent of all well-informed students are successful, that more than 85 per cent of all successful students are well-informed, that more than 95 per cent of all successful students are well-prepared, and that more than 95 per cent of all well-prepared students are successful. This probabilistic knowledge completed by further probabilistic estimations is given by the probabilistic knowledge-base $(\mathcal{B}, KB)$ in Fig. 1. $\mathcal{B}$ is the set of nodes $\{\mathsf{su}, \mathsf{wi}, \mathsf{wp}, \mathsf{pr}, \mathsf{st}, \mathsf{bo}, \mathsf{ex}, \mathsf{ho}\}$. KB is the least set of conditional constraints containing $(Y|X)[u_1, u_2]$ for each arrow from X to Y labeled with u_1, u_2.

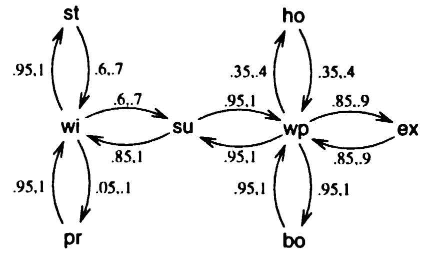


Figure 1: A Conditional Constraint Tree

Wondering, if it is useful for being successful at the university to interview the professors, to study on books, to spend the time on one's hobbies, or to do both studying on books and spending the time on one's hobbies, we get the probabilistic queries $\exists(\mathsf{su}|\mathsf{pr})[x_1, x_2]$, $\exists(\mathsf{su}|\mathsf{bo})[x_1, x_2]$, $\exists(\mathsf{su}|\mathsf{ho})[x_1, x_2]$, and $\exists(\mathsf{su}|\mathsf{bo\,ho})[x_1, x_2]$ yielding the tight answers $\{x_1/0.00,\ x_2/1.00\}$, $\{x_1/0.90,\ x_2/1.00\}$, $\{x_1/0.30,\ x_2/0.46\}$, and $\{x_1/0.71,\ x_2/1.00\}$, respectively. The deduced results are quite intuitive, since studying on books has a positive effect on the success of a student, while investing time in hobbies is acting in a negative way.

Wondering, if successful students at the university interviewed their professors, if they studied on books, if they spent their time with their hobbies, or if they both studied on books and spent their time with their hobbies, we get the probabilistic queries $\exists(\mathsf{pr}|\mathsf{su})[x_1, x_2]$, $\exists(\mathsf{bo}|\mathsf{su})[x_1, x_2]$, $\exists(\mathsf{ho}|\mathsf{su})[x_1, x_2]$, and $\exists(\mathsf{bo\,ho}|\mathsf{su})[x_1, x_2]$ yielding the tight answers $\{x_1/0.00,\ x_2/0.17\}$, $\{x_1/0.90,\ x_2/1.00\}$, $\{x_1/0.30,\ x_2/0.45\}$, and $\{x_1/0.25,\ x_2/0.45\}$, respectively. Again, the results make sense, since the biggest part of all successful students really studied, and just a small part of them spent their time with their hobbies.

Note that in this example, probabilistic interpretations of propositional events represent relative cardinalities of sets of objects. That is, more generally, in probabilistic propositional logics, we do not have the difference between probabilities on a set of possible worlds and those on a domain, like in probabilistic first-order logics (see, for example, [4] and [14]).

2.4 CONDITIONAL CONSTRAINT TREES

We formally define conditional constraint trees and queries to conditional constraint trees. We provide some additional examples, which are subsequently used as running examples.

A *(general) conditional constraint tree* is a probabilistic knowledge-base $(\mathcal{B}, KB)$ for which an undirected tree (a singly connected undirected graph) $(\mathcal{B}, \leftrightarrow)$ exists such that KB contains exactly one pair of conditional constraints $(B|A)[u_1, u_2]$ and $(A|B)[v_1, v_2]$ with $u_1, v_1 > 0$ for each pair of adjacent nodes A and B. A conditional constraint tree is *exact* iff $u_1 = u_2$ for all $(B|A)[u_1, u_2] \in KB$.

A *query to a conditional constraint tree* is a probabilistic query $\exists(F|E)[x_1, x_2]$ with conjunctive events E, F that are disjoint in their basic events and such that all paths from a basic event in E to a basic event in F have at least one basic event in common. A query $\exists(F|E)[x_1, x_2]$ to a conditional constraint tree is *premise-restricted* iff E is a basic event. It is *conclusion-restricted* iff F is a basic event. It is *complete* iff EF contains exactly the leaves of $(\mathcal{B}, \leftrightarrow)$.

Fig. 2 shows two conditional constraint trees of which the one on the left side is exact. $\exists$(GHI|ABEF)$[x_1, x_2]$ is a query, while $\exists$(AG|EI)$[x_1, x_2]$ is not a query to the conditional constraint trees of Fig. 2. Furthermore, $\exists$(GHI|A)$[x_1, x_2]$ is a premise-restricted query, $\exists$(C|EFGHI)$[x_1, x_2]$ a conclusion-restricted query, and $\exists$(EFGHI|A)$[x_1, x_2]$ a premise-restricted complete query to the conditional constraint trees of Fig. 2.

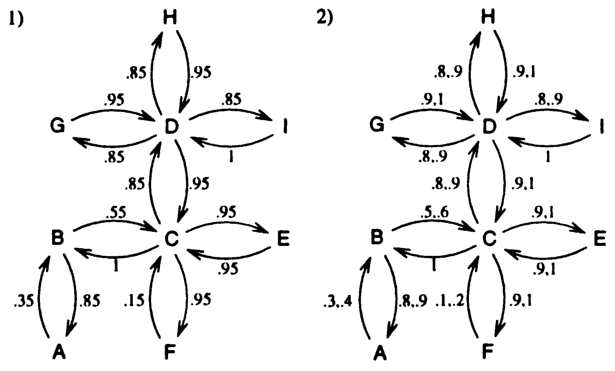


Figure 2: Two Conditional Constraint Trees

3 PROBABILISTIC DEDUCTION

The problem of computing the tight answer to a query to a conditional constraint tree can be divided into the subproblems of computing the tight answer to a premise-restricted query and the tight answer to a conclusion-restricted query.

For example, given the query $\exists$(GHI|ABEF)$[x_1, x_2]$ to the conditional constraint trees in Fig. 2, we first compute the tight answer $\{y_1/u_1,\ y_2/u_2\}$ to the premise-restricted query $\exists$(ABEF|C)$[y_1, y_2]$ and the tight answer $\{z_1/v_1,\ z_2/v_2\}$ to the conclusion-restricted query $\exists$(C|ABEF)$[z_1, z_2]$. We then generate a new conditional constraint tree by replacing the subtree over the nodes A, B, C, E, and F by the new pair of conditional constraints ('ABEF'|C)$[u_1, u_2]$ and (C|'ABEF')$[v_1, v_2]$ over the nodes 'ABEF' and C (in this new conditional constraint tree, ABEF acts as a new basic event 'ABEF'). Finally, we just compute the tight answer to the premise-restricted query $\exists$(GHI|'ABEF')$[x_1, x_2]$ to the new conditional constraint tree.

In the sequel, we restrict the considerations of this work to computing tight answers to premise-restricted queries. This subproblem reveals the main key ideas and the main theoretically interesting results of computing tight answers to queries. For more details on computing tight answers to queries, the interested reader may refer to our results in [19].

The problem of computing tight answers to premise-restricted queries can be reduced to the more specialized problem of calculating tight answers to premise-restricted complete queries. More precisely, for each premise-restricted query $\exists(F|E)[x_1, x_2]$ to a conditional constraint tree $(\mathcal{B}, KB)$, an equivalent premise-restricted complete query to a slightly transformed conditional constraint tree is generated as follows (see [19] for the correctness of this reduction):

In a first step, the conditional constraint tree is transformed by iteratively removing leaves not contained in EF, until all leaves are in EF. In a second transformation step, new basic events and new conditional constraints are added to the conditional constraint tree. For each non-leaf B contained in EF, $\mathcal{B}$ is increased by a new leaf B', and KB is increased by the new pair of conditional constraints $(B'|B)[1, 1]$ and $(B|B')[1, 1]$. The query is then transformed by replacing all non-leaves in E and F by the corresponding new leaves.

Hence, without loss of generality, we can restrict the subsequent considerations of this work to computing tight answers to premise-restricted complete queries.

3.1 EXACT CONDITIONAL CONSTRAINT TREES

In this section, let $(\mathcal{B}, KB)$ be an exact conditional constraint tree. To compute the tight answer to the premise-restricted complete query $\exists(F|E)[x_1, x_2]$, we start by defining a directed tree (that is, a directed acyclic graph in which each node has exactly one parent, except for the root that does not have any):

$A \dashrightarrow B$ iff $A \leftrightarrow B$ and A is closer to E than B.

The directed tree $(\mathcal{B}, \dashrightarrow)$ is uniquely determined by the conditional constraint tree and the premise-restricted complete query. Fig. 3 shows $(\mathcal{B}, \dashrightarrow)$ for the premise-restricted complete query $\exists$(EFGHI|A)$[x_1, x_2]$ to the exact conditional constraint tree in Fig. 2, left side.

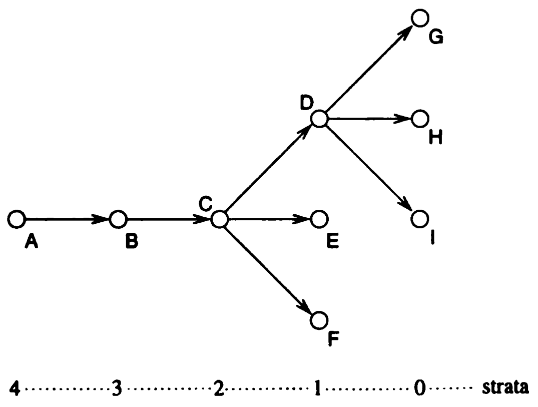


Figure 3: The Directed Tree $(\mathcal{B}, \dashrightarrow)$

The set of nodes $\mathcal{B}$ is partitioned into several strata. The lowest stratum contains only nodes with no children in $(\mathcal{B}, \dashrightarrow)$, the highest stratum contains the nodes with no parents in $(\mathcal{B}, \dashrightarrow)$ (that is, exactly the node of the premise E of the query). Fig. 3 also shows the different strata in our example.

At each node of $(\mathcal{B}, \dashrightarrow)$, we compute certain tightest bounds that are logically entailed by KB. More precisely, the tightest bounds at a node B are computed locally, by exploiting the tightest bounds that have been previously computed at the children of B. Hence, we iteratively compute the tightest bounds at the nodes of each stratum, starting with the nodes of the lowest stratum and terminating with the nodes of the highest stratum. We distinguish three different ways of computing tightest bounds at a node:

- initialization of a leaf (Leaf),
- chaining of an arrow and a subtree via a common node (Chaining),
- fusion of subtrees via a common node (Fusion).

Let us consider again the premise-restricted complete query $\exists$(EFGHI|A)$[x_1, x_2]$ to the exact conditional constraint tree in Fig. 2, left side. Fig. 4 illustrates the three different ways of computing tightest bounds at a node (the common nodes for Chaining and Fusion are filled black). Table 1 shows the greatest lower and the least upper bounds that are computed at each node B of each stratum st. More precisely, $\alpha_1 = \inf Pr(BC)/Pr(B)$, $\alpha_2 = \sup Pr(BC)/Pr(B)$, $\beta_2 = \sup Pr(\overline{B}C)/Pr(B)$, and $\gamma_2 = \sup Pr(C)/Pr(B)$ subject to $Pr \models KB$ and $Pr(B) > 0$ (in Table 1, we abbreviate Leaf, Chaining, and Fusion by Le, Ch, and Fu, respectively). Table 1 also shows the requested tight answer $\{x_1/0.02, x_2/0.17\}$, which is given by the tightest bounds computed at the premise A.

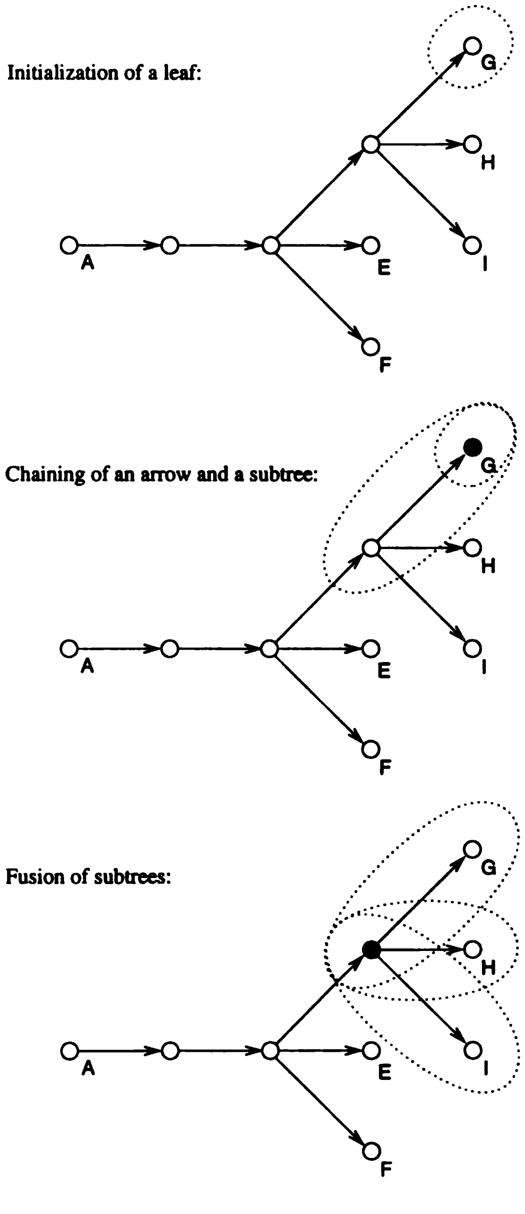

Initialization of a leaf:
A
E
F
G
H
I
Chaining of an arrow and a subtree:
A
E
F
G
H
I
Fusion of subtrees:
A
E
F
G
H
I

Figure 4: Local Computations in $(\mathcal{B}, \dashrightarrow)$

Table 1: Locally Computed Tightest Bounds

st	*B*	*C*	α_1	α_2	β_2	γ_2	
	G	G	1.0000	1.0000	0.0000	1.0000	(Le)
0	H	H	1.0000	1.0000	0.0000	1.0000	(Le)
	I	I	1.0000	1.0000	0.0000	1.0000	(Le)
	D	G	0.8500	0.8500	0.0447	0.8947	(Ch)
1	D	H	0.8500	0.8500	0.0447	0.8947	(Ch)
	D	I	0.8500	0.8500	0.0000	0.8500	(Ch)
	D	GHI	0.5500	0.8500	0.0000	0.8500	(Fu)
1	E	E	1.0000	1.0000	0.0000	1.0000	(Le)
	F	F	1.0000	1.0000	0.0000	1.0000	(Le)
	C	GHI	0.4474	0.7605	0.0447	0.7605	(Ch)
2	C	E	0.9500	0.9500	0.0500	1.0000	(Ch)
	C	F	0.9500	0.9500	5.3833	6.3333	(Ch)
2	C	EFGHI	0.3474	0.7605	0.0447	0.7605	(Fu)
3	B	EFGHI	0.1911	0.4183	0.0246	0.4183	(Ch)
4	A	EFGHI	0.0169	0.1722	0.0719	0.1722	(Ch)

Note that computing the requested tight answer by the classical linear programming approach would result in solving two linear programs, each of them having 512 variables and 34 linear inequalities.

Finally, we focus on the technical details. We present the functions P_1 and P_2, which compute the described greatest lower and least upper bounds. For this purpose, we need the following preparative definitions.

Let $P(BC,B) = u$ for all $(C|B)[u,u] \in KB$.

A node B is a *leaf* if it does not have any children. For all leaves B let $B^\uparrow = B$. For all the other nodes B let $B^\uparrow$ be the conjunction of all the children of B. For all leaves C let $L(C) = C$. For all the other conjunctive events C let $L(C)$ be the conjunction of all the leaves that are in C or that are descendants of a node in C.

In the sequel, let B be a node and let $C = B^\uparrow$. The case $C = B$ refers to the initialization of the leaf B, the case $C = B_1$ with a node $B_1 \neq B$ to the chaining of the arrow $B \rightarrow B_1$ and a subtree via the common node B_1, and the case $C = B_1 B_2 \ldots B_k$ with $k > 1$ nodes $B_1, B_2, \ldots, B_k$ to the fusion of k subtrees via the common node B.

We now define the function P_1 for the computation of greatest lower bounds. Note that its existence is already known from [27] in the framework of deductive database systems. Let $P_1(BL(C),B) = \alpha_1$, where α_1 in Leaf, Chaining, and Fusion is given as follows:

Leaf ($C = B$):

$$\alpha_1 = 1$$

Chaining ($C = B_1$):

$$\alpha_1 = \max(0, P(BC,B) \cdot (1 + \tfrac{P_1(CL(C^\uparrow),C)-1}{P(CB,C)}))$$

Fusion ($C = B_1 B_2 \ldots B_k$):

$$\alpha_1 = \max(0, 1 - k + \sum_{i=1}^{k} P_1(BL(B_i),B))$$

To formulate that P_1 computes greatest lower bounds, we need the following definitions. Let $\mathcal{B}(B,C)$ comprise B, all nodes in C and all descendants of a node in C. Let $KB(B,C)$ be the set of all conditional constraints of KB over $\mathcal{B}(B,C)$. Let $Mo(B,C)$ be the set of all models of $KB(B,C)$ over $\mathcal{B}(B,C)$.

Now, the function P_1 is sound and globally complete with respect to B and C iff $P_1(BL(C),B) = \alpha_1$ is the infimum of $Pr(BL(C)) \,/\, Pr(B)$ subject to $Pr \in Mo(B,C)$ and $Pr(B) > 0$. Thus, the next theorem shows soundness and global completeness of P_1.

Theorem 3.1

a) $\alpha_1 \cdot Pr(B) \leq Pr(BL(C))$ *for all* $Pr \in Mo(B,C)$.

b) There exists $Pr \in Mo(B,C)$ *with* $Pr(B) > 0$ *and* $\alpha_1 \cdot Pr(B) = Pr(BL(C))$.

Proof. The proof is given in full detail in [19]. □

Next, we present the function P_2 for computing least upper bounds. Note that it has been unknown in the literature so far if such a function exists. This function P_2 provides the crucial result that for exact conditional constraint trees, there are local probabilistic deduction techniques that are sound and globally complete. In detail, let $P_2(BL(C),B) = \alpha_2$, $P_2(\overline{B}L(C),B) = \beta_2$, and $P_2(L(C),B)) = \gamma_2$, where α_1, β_2, and γ_2 in Leaf, Chaining, and Fusion are given as follows:

Leaf ($C = B$):

$$(\alpha_2, \beta_2, \gamma_2) = (1, 0, 1)$$

Chaining ($C = B_1$):

$$\begin{aligned}\alpha_2 = \min(1, & P(BC,B) \cdot \tfrac{P_2(L(C^\uparrow),C)}{P(CB,C)}, \\ & 1 - P(BC,B) \cdot (1 - \tfrac{P_2(CL(C^\uparrow),C)}{P(CB,C)}), \\ & P(BC,B) \cdot (1 + \tfrac{P_2(\overline{C}L(C^\uparrow),C)}{P(CB,C)}))\end{aligned}$$

$$\begin{aligned}\beta_2 = \min(& P(BC,B) \cdot (\tfrac{P_2(\overline{C}L(C^\uparrow),C)+1}{P(CB,C)} - 1), \\ & P(BC,B) \cdot \tfrac{P_2(L(C^\uparrow),C)}{P(CB,C)})\end{aligned}$$

$$\gamma_2 = P(BC,B) \cdot \tfrac{P_2(L(C^\uparrow),C)}{P(CB,C)}$$

Fusion ($C = B_1 B_2 \ldots B_k$):

$$\alpha_2 = \min_{i \in [1:k]} P_2(BL(B_i),B)$$

$$\beta_2 = \min_{i \in [1:k]} P_2(\overline{B}L(B_i),B)$$

$$\begin{aligned}\gamma_2 \;=\; \min(&\min_{i\in[1:k]} P_2(L(B_i),B)\\ &\min_{i,j\in[1:k],i\neq j}(P_2(BL(B_i),B)+\\ &\qquad P_2(\overline{B}L(B_j),B)))\end{aligned}$$

The function P_2 is sound and globally complete with respect to B and C iff α_2, β_2, and γ_2 are the supremum of $Pr(BL(C))\,/\,Pr(B)$, $Pr(\overline{B}L(C))\,/\,Pr(B)$, and $Pr(L(C))\,/\,Pr(B)$, respectively, subject to $Pr \in Mo(B,C)$ and $Pr(B) > 0$. Hence, the following theorem shows soundness and global completeness of P_2 (actually, it shows even more to enable a proof by induction on the recursive definition of P_2).

Theorem 3.2

a) $Pr(BL(C)) \leq \alpha_2 \cdot Pr(B)$, $Pr(\overline{B}L(C)) \leq \beta_2 \cdot Pr(B)$, *and* $Pr(L(C)) \leq \gamma_2 \cdot Pr(B)$ *for all* $Pr \in Mo(B,C)$.

b) There exists $Pr \in Mo(B,C)$ *with* $Pr(B) > 0$, $Pr(BL(C)) = \alpha_2 \cdot Pr(B)$, *and* $Pr(L(C)) = \gamma_2 \cdot Pr(B)$.

c) There exists $Pr \in Mo(B,C)$ *with* $Pr(B) > 0$, $Pr(\overline{B}L(C)) = \beta_2 \cdot Pr(B)$, *and* $Pr(L(C)) = \gamma_2 \cdot Pr(B)$.

Proof. The proof is given in full detail in [19]. □

Note that Theorem 3.2 also shows that $P_2(L(B_i),B) \leq P_2(BL(B_i),B)+P_2(\overline{B}L(B_i),B)$ for all $i \in [1:k]$. Thus, $\min_{i,j\in[1:k],i\neq j}(P_2(BL(B_i),B)+P_2(\overline{B}L(B_j),B))$ in the definition of γ_2 in FUSION can be replaced by $\alpha_2+\beta_2$ for an increased efficiency in computing γ_2 by exploiting the already computed values of α_2 and β_2.

Briefly, by Theorems 3.1 and 3.2, the tight answer to the premise-restricted complete query $\exists(F|E)[x_1,x_2]$ is $\{x_1/P_1(EL(E^\dagger),E),\ x_2/P_2(EL(E^\dagger),E)\}$.

The time complexity of computing the tight answer to a premise-restricted complete query this way, by using the functions P_1 and P_2, is analyzed in Section 3.3.

3.2 CONDITIONAL CONSTRAINT TREES

In this section, we focus on computing the tight answer to premise-restricted complete queries to general conditional constraint trees. In the sequel, let $(\mathcal{B},KB)$ be a conditional constraint tree and let $\exists(F|E)[x_1,x_2]$ be a premise-restricted complete query.

We may think that our local deduction technique for exact conditional constraint trees of the previous section can easily be generalized to conditional constraint trees. In fact, this is true as far as the computation of greatest lower bounds is concerned. However, the computation of least upper bounds cannot be generalized that easily from exact conditional constraint trees to conditional constraint trees. More precisely, generalizing the computation of least upper bounds results in solving global nonlinear programs. These nonlinear programs and our way to solve them are illustrated by the following example.

Let the conditional constraint tree $(\mathcal{B},KB)$ be given by $\mathcal{B}=\{\mathsf{A},\mathsf{B},\mathsf{C},\mathsf{D}\}$ and $KB=\{(\mathsf{B}|\mathsf{A})[u_1,u_2], (\mathsf{A}|\mathsf{B})[v_1,v_2], (\mathsf{C}|\mathsf{B})[x_1,x_2], (\mathsf{B}|\mathsf{C})[y_1,y_2], (\mathsf{D}|\mathsf{C})[r_1,r_2], (\mathsf{C}|\mathsf{D})[s_1,s_2]\}$, and let the premise-restricted complete query be defined by $\exists(\mathsf{D}|\mathsf{A})[z_1,z_2]$.

The requested least upper bound is the maximum of z subject to $u \in [u_1,u_2]$, $x \in [x_1,x_2]$, and the following nonlinear inequalities in (1) to (5):

$$\begin{array}{lrcl}
(1) & z & \leq & 1\\
(2) & z & \leq & 1-u+\frac{u}{v_1}-\frac{ux}{v_1}+\frac{uxr_2}{v_1y_1}\\
(3) & z & \leq & 1-u+\frac{ux}{v_1}-\frac{uxr_2}{v_1y_1}+\frac{uxr_2}{v_1y_1s_1}\\
(4) & z & \leq & u-\frac{ux}{v_1}+\frac{ux}{v_1y_1}-\frac{uxr_2}{v_1y_1}+\frac{uxr_2}{v_1y_1s_1}\\
(5) & z & \leq & \frac{uxr_2}{v_1y_1s_1}
\end{array}$$

We transform this nonlinear program into an equivalent linear program as follows. The maximum of z subject to $u \in [u_1,u_2]$, $x \in [x_1,x_2]$, and the nonlinear inequalities in (1) to (5) is equal to the maximum of z subject to the following system of linear inequalities over z and $x_B \geq 0$ $(B \in \mathcal{B})$:

$$\begin{array}{rcllllllll}
z & \leq & x_{\mathsf{A}} & & & & & & & \\
z & \leq & x_{\mathsf{A}} & + & \frac{1-v_1}{v_1}\cdot x_{\mathsf{B}} & - & \frac{y_1}{v_1y_1}\cdot x_{\mathsf{C}} & + & \frac{s_1}{v_1y_1s_1}\cdot x_{\mathsf{D}}\\
z & \leq & x_{\mathsf{A}} & - & \frac{v_1}{v_1}\cdot x_{\mathsf{B}} & + & \frac{y_1}{v_1y_1}\cdot x_{\mathsf{C}} & + & \frac{1-s_1}{v_1y_1s_1}\cdot x_{\mathsf{D}}\\
z & \leq & & & \frac{v_1}{v_1}\cdot x_{\mathsf{B}} & + & \frac{1-y_1}{v_1y_1}\cdot x_{\mathsf{C}} & + & \frac{1-s_1}{v_1y_1s_1}\cdot x_{\mathsf{D}}\\
z & \leq & & & & & & & \frac{1}{v_1y_1s_1}\cdot x_{\mathsf{D}}
\end{array}$$

$$\begin{array}{rcl}
1 \leq x_{\mathsf{A}} \leq 1, & & u_1\cdot x_{\mathsf{A}} \leq x_{\mathsf{B}} \leq u_2\cdot x_{\mathsf{A}}\\
x_1\cdot x_{\mathsf{B}} \leq x_{\mathsf{C}} \leq x_2\cdot x_{\mathsf{B}}, & & r_1\cdot x_{\mathsf{C}} \leq x_{\mathsf{D}} \leq r_2\cdot x_{\mathsf{C}}
\end{array}$$

More generally, least upper bounds to premise-restricted complete queries to conditional constraint trees can be expressed by similar nonlinear programs, which can similarly be transformed into linear programs.

For example, let us consider the premise-restricted complete query $\exists(\mathsf{EFGHI}|\mathsf{A})[x_1,x_2]$ to the conditional constraint tree in Fig. 2, right side. The requested least upper bound is the maximum of x subject to the system of linear inequalities in Fig. 5 (note that we actually generated 72 linear inequalities of which 31 were trivially subsumed by others).

Thus, in this example, the requested least upper bound is computed by solving a linear program that has 10 variables and 72 linear inequalities.

$$
\begin{array}{rcl@{\qquad}rcl}
x & \le & x_{\mathsf{A}} & x & \le & \frac{5}{4}x_{\mathsf{C}} + \frac{5}{36}x_{\mathsf{E}} \\
x & \le & x_{\mathsf{A}} + \frac{1}{4}x_{\mathsf{B}} - \frac{5}{4}x_{\mathsf{C}} + \frac{5}{4}x_{\mathsf{D}} & x & \le & \frac{5}{4}x_{\mathsf{C}} + \frac{45}{4}x_{\mathsf{F}} \\
x & \le & x_{\mathsf{A}} + \frac{1}{4}x_{\mathsf{B}} - \frac{5}{4}x_{\mathsf{C}} + \frac{5}{4}x_{\mathsf{E}} & x & \le & \frac{5}{4}x_{\mathsf{D}} + \frac{5}{36}x_{\mathsf{E}} \\
x & \le & x_{\mathsf{A}} + \frac{1}{4}x_{\mathsf{B}} - \frac{5}{4}x_{\mathsf{C}} + \frac{5}{4}x_{\mathsf{F}} & x & \le & \frac{5}{36}x_{\mathsf{D}} + \frac{5}{4}x_{\mathsf{E}} \\
x & \le & x_{\mathsf{A}} + \frac{1}{4}x_{\mathsf{B}} - \frac{5}{4}x_{\mathsf{D}} + \frac{25}{18}x_{\mathsf{G}} & x & \le & \frac{5}{36}x_{\mathsf{D}} + \frac{5}{4}x_{\mathsf{F}} \\
x & \le & x_{\mathsf{A}} + \frac{1}{4}x_{\mathsf{B}} - \frac{5}{4}x_{\mathsf{D}} + \frac{25}{18}x_{\mathsf{H}} & x & \le & \frac{5}{36}x_{\mathsf{E}} + \frac{5}{4}x_{\mathsf{F}} \\
x & \le & x_{\mathsf{A}} + \frac{1}{4}x_{\mathsf{B}} - \frac{5}{4}x_{\mathsf{D}} + \frac{25}{18}x_{\mathsf{I}} & x & \le & \frac{25}{18}x_{\mathsf{E}} \\
x & \le & x_{\mathsf{A}} - x_{\mathsf{B}} + \frac{5}{4}x_{\mathsf{C}} + \frac{5}{36}x_{\mathsf{F}} & x & \le & \frac{25}{2}x_{\mathsf{F}} \\
x & \le & x_{\mathsf{B}} + \frac{1}{9}x_{\mathsf{D}} & x & \le & \frac{25}{18}x_{\mathsf{G}} \\
x & \le & x_{\mathsf{B}} + \frac{1}{9}x_{\mathsf{E}} & x & \le & \frac{25}{18}x_{\mathsf{H}} \\
x & \le & x_{\mathsf{B}} + \frac{1}{9}x_{\mathsf{F}} & x & \le & \frac{25}{18}x_{\mathsf{I}} \\
x & \le & \frac{5}{4}x_{\mathsf{C}} + \frac{5}{36}x_{\mathsf{D}} & & &
\end{array}
$$

$$1 \le x_{\mathsf{A}} \le 1$$

$$
\begin{array}{rcccl@{\qquad}rcccl}
\frac{3}{10}x_{\mathsf{A}} & \le & x_{\mathsf{B}} & \le & \frac{2}{5}x_{\mathsf{A}}, & \frac{1}{2}x_{\mathsf{B}} & \le & x_{\mathsf{C}} & \le & \frac{3}{5}x_{\mathsf{B}} \\
\frac{9}{10}x_{\mathsf{C}} & \le & x_{\mathsf{F}} & \le & x_{\mathsf{C}}, & \frac{9}{10}x_{\mathsf{C}} & \le & x_{\mathsf{E}} & \le & x_{\mathsf{C}} \\
\frac{4}{5}x_{\mathsf{C}} & \le & x_{\mathsf{D}} & \le & \frac{9}{10}x_{\mathsf{C}}, & \frac{4}{5}x_{\mathsf{D}} & \le & x_{\mathsf{G}} & \le & \frac{9}{10}x_{\mathsf{D}} \\
\frac{4}{5}x_{\mathsf{D}} & \le & x_{\mathsf{H}} & \le & \frac{9}{10}x_{\mathsf{D}}, & \frac{4}{5}x_{\mathsf{D}} & \le & x_{\mathsf{I}} & \le & \frac{9}{10}x_{\mathsf{D}}
\end{array}
$$

Figure 5: The Generated Linear Inequalities

Note that computing the requested least upper bound by the classical linear programming approach would result in solving a linear program that has 512 variables and 34 linear inequalities.

Let us now focus on the technical details. We subsequently generalize the function P_1 of Section 3.1 in a straightforward way to compute greatest lower bounds in conditional constraint trees. Moreover, we present a linear program for computing the requested least upper bound in conditional constraint trees.

Let $P_1(BC, B) = u_1$ for all $(C|B)[u_1, u_2] \in KB$.

In the sequel, let B be a node and let $C = B^{\uparrow}$. Like in Section 3.1, $C = B$, $C = B_1$ with a node $B_1 \neq B$, and $C = B_1 B_2 \ldots B_k$ with $k > 1$ nodes $B_1, B_2, \ldots, B_k$ refer to LEAF, CHAINING, and FUSION, respectively.

We define the generalized function P_1 for computing greatest lower bounds in conditional constraint trees. Note that its existence is already known from [27] in the framework of deductive database systems. Let $P_1(BL(C), B) = \alpha_1$, where α_1 in LEAF, CHAINING, and FUSION is given as follows:

LEAF ($C = B$):

$$\alpha_1 = 1$$

CHAINING ($C = B_1$):

$$\alpha_1 = \max(0, P_1(BC, B) \cdot (1 + \tfrac{P_1(CL(C^{\uparrow}), C) - 1}{P_1(CB, C)}))$$

FUSION ($C = B_1 B_2 \ldots B_k$):

$$\alpha_1 = \max(0, 1 - k + \sum_{i=1}^{k} P_1(BL(B_i), B))$$

The function P_1 is sound and globally complete with respect to B and C iff $P_1(BL(C), B) = \alpha_1$ is the infimum of $Pr(BL(C)) \,/\, Pr(B)$ subject to $Pr \in Mo(B, C)$ and $Pr(B) > 0$. Thus, the next theorem shows soundness and global completeness of P_1.

Theorem 3.3

a) $\alpha_1 \cdot Pr(B) \le Pr(BL(C))$ *for all* $Pr \in Mo(B, C)$.

b) There exists $Pr \in Mo(B, C)$ *with* $Pr(B) > 0$ *and* $\alpha_1 \cdot Pr(B) = Pr(BL(C))$.

Proof. The claims follow from Theorem 3.1. □

Next, we concentrate on computing the requested least upper bound. More precisely, we formulate a linear program similar to the ones described in the two examples. We do this by defining the function I over the variables x_B ($B \in \mathcal{B}$). Each admissible instantiation of the variables corresponds to an exact conditional constraint tree as instance of the considered conditional constraint tree. Given such an admissible instantiation of the variables, I now provides least upper bounds for the corresponding exact conditional constraint tree. Hence, to obtain least upper bounds for the conditional constraint tree, we must maximize the value of I subject to all admissible instantiations of the variables. In detail, let $I(BL(C), B) = \alpha_2$, $I(\overline{B}L(C), B) = \beta_2$, and $I(L(C), B)) = \gamma_2$, where α_1, β_2, and γ_2 in LEAF, CHAINING, and FUSION are given as follows:

LEAF ($C = B$):

$$(\alpha_2, \beta_2, \gamma_2) = (x_B, 0, x_B)$$

CHAINING ($C = B_1$):

$$\alpha_2 = \min(x_B, \tfrac{I(L(C^{\uparrow}), C)}{P_1(CB, C)}, x_C + \tfrac{I(\overline{C}L(C^{\uparrow}), C)}{P_1(CB, C)}, x_B - x_C + \tfrac{I(CL(C^{\uparrow}), C)}{P_1(CB, C)})$$

$$\beta_2 = \min(\tfrac{1 - P_1(CB, C)}{P_1(CB, C)} \cdot x_C + \tfrac{I(\overline{C}L(C^{\uparrow}), C)}{P_1(CB, C)}, \tfrac{I(L(C^{\uparrow}), C)}{P_1(CB, C)})$$

$$\gamma_2 = \tfrac{I(L(C^{\uparrow}), C)}{P_1(CB, C)}$$

FUSION ($C = B_1 B_2 \ldots B_k$):

$$\alpha_2 = \min_{i \in [1:k]} I(BL(B_i), B)$$

$$\beta_2 = \min_{i\in[1:k]} I(\overline{B}L(B_i), B)$$

$$\gamma_2 = \min(\min_{i\in[1:k]} I(L(B_i), B), \min_{i,j\in[1:k], i\neq j} (I(BL(B_i), B) + I(\overline{B}L(B_j), B)))$$

It remains to specify the admissibility of an instantiation of the variables. This is done by the system of linear inequalities $J(E)$, which is defined as the least set of linear inequalities over $x_B \geq 0$ $(B \in \mathcal{B})$ that contains $1 \leq x_E \leq 1$ and $u_1 \cdot x_B \leq x_C \leq u_2 \cdot x_B$ for all $(C|B)[u_1, u_2] \in KB$ with $B \to C$ (that is, B is the parent of C). Note that $J(E)$ is just related to the final computation of the requested least upper bound.

We implicitly performed the variable transformation described in the two examples. The next lemma shows that this variable transformation is indeed correct for conditional constraint trees.

Lemma 3.4

a) If x_B $(B \in \mathcal{B})$ satisfies $J(E)$, then for all conditional constraints $(C|B)[u_1, u_2] \in KB$ such that $B \to C$, there exists $u_C \in [u_1, u_2]$ with $x_C = u_C \cdot x_B$.

b) Let $u_C \in [u_1, u_2]$ for all $(C|B)[u_1, u_2] \in KB$ such that $B \to C$. There exists x_B $(B \in \mathcal{B})$ with $J(E)$ and $x_C = u_C \cdot x_B$ for all nodes C with parent B.

Proof. a) For all nodes C with parent B, let u_C be defined by $u_C = x_C \,/\, x_B$.

b) Let $x_E = 1$, and for all nodes C with parent B, let x_C be defined by $x_C = u_C \cdot x_B$. □

We are now ready to formulate an optimization problem for computing the requested least upper bound.

Theorem 3.5 *Let X_2 be the maximum of x subject to $x \leq I(EL(E^\dagger), E)$ and $J(E)$.*

a) $Pr(EL(E^\dagger)) \leq X_2 \cdot Pr(E)$ for all $Pr \in Mo(E, E^\dagger)$.

b) There exists $Pr \in Mo(E, E^\dagger)$ with $Pr(E) > 0$ and $Pr(EL(E^\dagger)) = X_2 \cdot Pr(E)$.

Proof. Let $P(CB, C) = v_1$ for all $(B|C)[v_1, v_2] \in KB$ such that $B \to C$. By Theorem 3.2, the requested least upper bound is the maximum of x subject to $x \leq P_2(EL(E^\dagger), E)$ and $P(BC, B) = u_C \in [u_1, u_2]$ for all $(C|B)[u_1, u_2] \in KB$ such that $B \to C$. By Lemma 3.4, we can equivalently maximize x subject to $x \leq I(EL(E^\dagger), E)$ and $J(E)$. □

We wonder how to solve the generated optimization problem, since $I(EL(E^\dagger), E)$ may still contain min-operations that cannot be tackled by linear programming. Given a technique for solving this optimization problem, we are also interested in a rough idea on the overall time complexity of computing the requested least upper bound this way. Moreover, we are interested in possible improvements to increase efficiency. These topics are discussed in the following.

If $I(EL(E^\dagger), E)$ does not contain any min-operations at all, then the generated optimization problem is already a linear program. Otherwise, it can easily be transformed into a linear program. In a first transformation step, all inner min-operations are eliminated. This can easily be done due to the well-structuredness of $I(EL(E^\dagger), E)$. In a second step, the only remaining outer min-operation is eliminated by introducing exactly one linear inequality for each contained operand. In these linear inequalities, the operands of the outer min-operation are upper bounds of x.

To get a rough idea on the time complexity of computing the requested least upper bound this way, we must analyze the size of the generated linear programs. It is given by the number of variables, the number of linear inequalities in $J(E)$, and the number of linear inequalities extracted from $x \leq I(EL(E^\dagger), E)$. The latter number is quite worrying, since γ_2 seems to produce many min-operands in FUSION. Moreover, γ_2 contains $I(BL(B_i), B)$ in FUSION, and α_2 contains $I(L(C^\dagger), C)$ in CHAINING. So, due to this crossed dependency, the overall number of generated linear inequalities is likely to 'explode' for trees that branch very often.

In order to avoid these problems, we now introduce the auxiliary function I' over the variables x_B $(B \in \mathcal{B})$. Let $I'(BL(C), B) = \alpha'_2$, $I'(\overline{B}L(C), B) = \beta'_2$, and $I'(L(C), B)) = \gamma'_2$, where α'_1, β'_2, and γ'_2 in LEAF, CHAINING, and FUSION are given as follows:

LEAF $(C = B)$:

$$(\alpha'_2, \beta'_2, \gamma'_2) = (x_B, 0, x_B)$$

CHAINING $(C = B_1)$:

$$\alpha'_2 = \min(x_B,\, x_C + \tfrac{I'(\overline{C}L(C^\dagger), C)}{P_1(CB, C)},\; x_B - x_C + \tfrac{I'(CL(C^\dagger), C)}{P_1(CB, C)})$$

$$\beta'_2 = \tfrac{1 - P_1(CB, C)}{P_1(CB, C)} \cdot x_C + \tfrac{I'(\overline{C}L(C^\dagger), C)}{P_1(CB, C)}$$

$$\gamma'_2 = \tfrac{I'(L(C^\dagger), C)}{P_1(CB, C)}$$

FUSION $(C = B_1 B_2 \ldots B_k)$:

$$\alpha'_2 = \min_{i\in[1:k]} I'(BL(B_i), B)$$

$$\beta'_2 = \min_{i\in[1:k]} I'(\overline{B}L(B_i), B)$$

$$\gamma_2' = \min(\min_{i\in[1:k]} I'(L(B_i),B), \min_{i,j\in[1:k],i\neq j}(I'(BL(B_i),B)+ I'(\overline{B}L(B_j),B)))$$

Note that in the definition of α_2' in CHAINING, we can consider the cases $C^\uparrow = C$ and $C^\uparrow \neq C$ separately. Since simply $\alpha_2' = x_C$ for $C^\uparrow = C$, we reduce the number of generated linear inequalities this way.

The next lemma shows that the function I can be expressed in terms of the auxiliary function I'.

Lemma 3.6 *For all x_B ($B \in \mathcal{B}$) that satisfy $J(E)$:*

$$\alpha_2 = \min(\alpha_2', \gamma_2'),\ \beta_2 = \min(\beta_2', \gamma_2'),\ \textit{and}\ \gamma_2 = \gamma_2'.$$

Proof sketch. The claim can be proved by induction on the recursive definition of I. □

Briefly, by Theorem 3.3, Theorem 3.5, and Lemma 3.6, the tight answer to the premise-restricted complete query $\exists(F|E)[x_1,x_2]$ is $\{x_1/P_1(EL(E^\uparrow),E),\ x_2/X_2\}$, where X_2 is given by the maximum of x subject to $x \leq I'(EL(E^\uparrow),E)$, $x \leq I'(L(E^\uparrow),E)$, and $J(E)$.

In our example, we get $\{x_1/0.00,\ x_2/0.27\}$ as the tight answer to the query $\exists(\mathsf{EFGHI}|\mathsf{A})[x_1,x_2]$ to the conditional constraint tree in Fig. 2, right side.

The time complexity of this way of computing the requested greatest lower bound and especially the requested least upper bound is analyzed in Section 3.3.

3.3 COMPLEXITY

For exact conditional constraint trees, our approach to compute the tight answer in several strata by the functions P_1 and P_2 runs in time $O(n)$, where n denotes the number of nodes. This result is a consequence of the following observations. The directed tree can be computed in time $O(n)$. An initialization of a leaf with a constant number of assignments is performed exactly for each leaf of the directed tree, a chaining with a constant number of arithmetic operations is performed exactly for each arrow of the directed tree. Hence, initializing all leaves and performing all chainings runs in time $O(n)$. A fusion is done for each branching of the directed tree, using linear time in the number of branches. Thus, all fusions together run in time $O(n)$.

For general conditional constraint trees, the requested greatest lower bound is analogously computed in time $O(n)$. The requested least upper bound is computed in polynomial time in n. This result follows from the fact that linear programming runs in polynomial time in the size of the linear programs (see, for example, [24] and [26]). The size of the generated linear programs is given by the number of variables and the number of linear inequalities. The number of variables is $n+1$. The number of linear inequalities in $J(E)$ is $2n$. By induction on the recursive definition of I', it can be shown that the number of min-operands in α_2', β_2', and γ_2' is limited by $|\mathcal{B}(B,C)|^2$, $|\mathcal{B}(B,C)|$, and $|\mathcal{B}(B,C)|^4$, respectively. Hence, the number of linear inequalities extracted from $x \leq I'(EL(E^\uparrow),E)$ and $x \leq I'(L(E^\uparrow),E)$ is limited by $|\mathcal{B}(E,E^\uparrow)|^2 + |\mathcal{B}(E,E^\uparrow)|^4 = n^2+n^4$. Thus, the overall number of generated linear inequalities l is limited by $l_u = 2n + n^2 + n^4$.

Note that l_u is a very rough upper bound for l, in many conditional constraint trees (especially in those that branch very rarely), l is much lower than l_u. For example, taking a complete binary tree with $n = 127$ nodes, we get only $l = 19\,964$ compared to $l_u = 260\,161\,024$. In the example of Section 3.2 with $n = 9$ nodes, we get only $l = 72$ compared to $l_u = 6\,660$. Another example is a tree that is degenerated to a chain of basic events. In this case, we get $l = 5n + 1$, that is, the overall number of generated linear inequalities is linear in n.

As a comparison, we shortly focus on the classical linear programming approach to probabilistic deduction. It runs in exponential time in n. In the special case of computing tight answers to queries to exact and to general conditional constraint trees, two linear programs are to be solved, each of them having 2^n variables and $4n - 2$ linear inequalities.

4 SUMMARY AND CONCLUSIONS

We showed that probabilistic deduction with conditional constraints over basic events is NP-hard. We then presented very efficient and globally complete techniques for probabilistic deduction in conditional constraint trees. For exact conditional constraint trees, greatest lower and least upper bounds are deduced in linear time in the size of the conditional constraint trees. For general conditional constraint trees, greatest lower bounds are deduced in the same way, in linear time in the size of the conditional constraint trees. However, computing least upper bounds seems to be computationally more difficult. It is done by solving global nonlinear programs. We showed how to transform these nonlinear programs into equivalent linear programs, which are solvable in polynomial time in the size of the conditional constraint trees.

Conditional constraint trees and the analyzed probabilistic deduction problems related to them seem to be very interesting for many practical applications. Like Bayesian networks, they represent well-structured probabilistic knowledge-bases, which have an intuitive graphical representation. However, differently from Bayesian networks, conditional constraint trees do not require any probabilistic independence assumptions. So, rather than being in competition with Bayesian networks, conditional constraint trees are to be understood as a complement to Bayesian networks, useful for applications in which well-structured independence assumptions do not hold or are difficult to access.

Acknowledgements

I am very grateful to Thomas Eiter for valuable comments on an earlier version of this paper.

References

[1] E. W. Adams. *The Logic of Conditionals*, volume 86 of *Synthese Library*. D. Reidel Publishing Company, Dordrecht, Holland, 1975.

[2] S. Amarger, D. Dubois, and H. Prade. Constraint propagation with imprecise conditional probabilities. In *Proc. of the 7th Conference on Uncertainty in Artificial Intelligence*, pages 26–34. Morgan Kaufmann Publishers, 1991.

[3] K. A. Andersen and J. N. Hooker. Bayesian logic. *Decision Support Systems*, 11:191–210, 1994.

[4] F. Bacchus. *Representing and Reasoning with Probabilistic Knowledge: A Logical Approach to Probabilities.* The MIT Press, Cambridge, USA, 1990.

[5] R. Carnap. *Logical Foundations of Probability.* University of Chicago Press, Chicago, 1950.

[6] B. de Finetti. *Theory of Probability.* Wiley, New York, 1974.

[7] D. Dubois and H. Prade. On fuzzy syllogisms. *Computational Intelligence*, 4(2):171–179, 1988.

[8] D. Dubois, H. Prade, L. Godo, and R. L. de Màntaras. A symbolic approach to reasoning with linguistic quantifiers. In *Proc. of the 8th Conference on Uncertainty in Artificial Intelligence*, pages 74–82. Morgan Kaufmann Publishers, 1992.

[9] D. Dubois, H. Prade, and J.-M. Touscas. Inference with imprecise numerical quantifiers. In Z. W. Ras and M. Zemankova, editors, *Intelligent Systems*, chapter 3, pages 53–72. Ellis Horwood, 1990.

[10] R. Fagin, J. Y. Halpern, and N. Megiddo. A logic for reasoning about probabilities. *Information and Computation*, 87:78–128, 1990.

[11] A. M. Frisch and P. Haddawy. Anytime deduction for probabilistic logic. *Artificial Intelligence*, 69:93–122, 1994.

[12] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman & Co., 1979.

[13] G. Georgakopoulos, D. Kavvadias, and C. H. Papadimitriou. Probabilistic satisfiability. *Journal of Complexity*, 4(1):1–11, 1988.

[14] J. Y. Halpern. An analysis of first-order logics of probability. *Artifical Intelligence*, pages 311–350, 1990.

[15] J. Heinsohn. Probabilistic description logics. In *Proc. of the 10th Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers, 1994.

[16] B. Jaumard, P. Hansen, and M. P. de Aragão. Column generation methods for probabilistic logic. *ORSA Journal of Computing*, 3:135–147, 1991.

[17] D. Kavvadias and C. H. Papadimitriou. A linear programming approach to reasoning about probabilities. *Annals of Mathematics and Artificial Intelligence*, pages 189–205, 1990.

[18] T. Lukasiewicz. Uncertain reasoning in concept lattices. In *Proc. of the 3rd European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, volume 946 of *LNCS/LNAI*, pages 293–300. Springer, 1995.

[19] T. Lukasiewicz. *Precision of Probabilistic Deduction under Taxonomic Knowledge.* Doctoral Dissertation, Universität Augsburg, 1996.

[20] T. Lukasiewicz. Efficient global probabilistic deduction from taxonomic and probabilistic knowledge-bases over conjunctive events. In *Proc. of the 6th International Conference on Information and Knowledge Management*, pages 75–82. ACM Press, 1997.

[21] T. Lukasiewicz, W. Kießling, G. Köstler, and U. Güntzer. Taxonomic and uncertain integrity constraints in object-oriented databases - the TOP approach. In *Proc. of the 4th International Conference on Information and Knowledge Management*, pages 241–249. ACM Press, 1995.

[22] C. Luo, C. Yu, J. Lobo, G. Wang, and T. Pham. Computation of best bounds of probabilities from uncertain data. *Computational Intelligence*, 12(4):541–566, 1996.

[23] N. J. Nilsson. Probabilistic logic. *Artificial Intelligence*, 28:71–88, 1986.

[24] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization, Algorithms and Complexity.* Prentice-Hall, Englewood Cliffs, New Jersey, 1982.

[25] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers, San Mateo, California, 1988.

[26] A. Schrijver. *Theory of Linear and Integer Programming.* Wiley, New York, 1986.

[27] H. Thöne. *Precise Conclusion under Uncertainty and Incompleteness in Deductive Database Systems.* Doctoral Dissertation, Universität Tübingen, 1994.

[28] H. Thöne, U. Güntzer, and W. Kießling. Towards precision of probabilistic bounds propagation. In *Proc. of the 8th Conf. on Uncertainty in Artificial Intelligence*, pages 315–322. Morgan Kaufmann Publishers, 1992.

Logic Programming–based Representations II

Specifying Transactions for Extended Abduction

Katsumi Inoue
Dept. Electrical and Electronics Engineering
Kobe University
Rokkodai, Nada, Kobe 657-8501, Japan
inoue@eedept.kobe-u.ac.jp

Chiaki Sakama
Dept. Computer and Communication Sciences
Wakayama University
Sakaedani, Wakayama 640-8441, Japan
sakama@sys.wakayama-u.ac.jp

Abstract

Extended abduction introduced by Inoue and Sakama (1995) generalizes traditional abduction in the sense that it can compute *negative explanations* by removing hypotheses from a nonmonotonic background theory, rather than only adding them. Also, it has a mechanism of computing *anti-explanations* to *un*explain negative observations. Such extended abduction not only enhances reasoning ability of traditional abduction but has useful applications to nonmonotonic theory change. In this paper, we study the computational aspect of extended abduction. Given a background theory written in nonmonotonic logic programming, we introduce its *transaction program* for computing extended abduction. A transaction program is a set of nondeterministic production rules that specify addition and deletion of abductive hypotheses. Abductive explanations are computed by the fixpoint of a transaction program using a bottom-up model generation procedure. In the context of databases, a transaction program provides a declarative specification of database update.

1 INTRODUCTION

Given a background knowledge base K and an observation G, abduction computes an explanation E of G satisfying

$$K \cup E \models G$$

where $K \cup E$ is consistent. This is the traditional logical framework of abduction widely accepted in AI (Poole, 1988). However, the above framework of abduction is not sufficient when the background knowledge base is *nonmonotonic*. For example, consider the knowledge base written in nonmonotonic logic programming:

$$\begin{aligned} K: \quad & flies(x) \leftarrow bird(x), not\, ab(x), \\ & ab(x) \leftarrow broken\text{-}wing(x), \\ & bird(tweety) \leftarrow , \\ & bird(opus) \leftarrow , \\ & broken\text{-}wing(tweety) \leftarrow . \end{aligned}$$

If we observe that *tweety* flies, there is a good reason to assume that the wound has already healed. Then, the fact *broken-wing*(*tweety*) must be *removed* from the knowledge base to explain the observation *flies*(*tweety*).

The traditional abduction cannot characterize such a situation. That is, abduction introduces hypotheses to a knowledge base, but once they are included, any hypothesis cannot be removed from the knowledge base. To cope with this problem, Inoue and Sakama (1995) introduced the notion of "negative explanations". Given a background knowledge base K and an observation G, a set F of formulas is called a *negative explanation* of G if

$$K \setminus F \models G$$

where $K \setminus F$ is consistent. An explanation E satisfying $K \cup E \models G$ is then called a *positive explanation*.

On the other hand, suppose that we later notice that *opus* does not fly any more. Since *flies*(*opus*) is entailed by K, we now have to revise the knowledge base to block the derivation of *flies*(*opus*) by assuming *broken-wing*(*opus*), for instance. This situation is characterized by the notion of "anti-explanations" which are used to *un*explain observations (Inoue and Sakama, 1995). Given a background knowledge base K and an observation G, a set E of formulas is called a *(positive) anti-explanation* of G if

$$K \cup E \not\models G,$$

and a set F of formulas is called a *negative anti-explanation* of G if

$$K \setminus F \not\models G.$$

These extensions of traditional abduction are useful when we consider nonmonotonic theories as background knowledge bases. Firstly, the introduction of negative explanations is natural, since it is often the case that deletion of formulas introduces new formulas in nonmonotonic reasoning. In fact, positive and negative explanations play a complementary role in accounting for an observation in nonmonotonic theories. Secondly, the introduction of anti-explanations is necessary when one wants to account for *negative observations*. In this respect, traditional abduction is concerned with explaining *positive observations* only. Negative observations are often perceived in real-life situations. For instance, many *inductive concept learning* systems consider both positive and negative examples to construct a hypothetical theory. In this sense, the notion of anti-explanations plays a dual role to explanations. Thirdly, extended abduction not only enhances reasoning ability of traditional abduction, but is essential to theories of *nonmonotonic theory change* and *abductive theory revision* (Inoue and Sakama, 1995). Useful applications of extended abduction thus include *view update* in deductive databases, *contradiction removal* as well as *diagnosis* and *repair*. For example, model-based diagnosis (Reiter, 1987) constructs a diagnostic hypothesis by identifying a minimal set of fault components Δ from the set of system components $COMP$ such that (i)

$$H = \{Ab(c) \mid c \in \Delta\} \cup \{\neg Ab(c) \mid c \in COMP \setminus \Delta\}$$

and (ii) H is consistent with the system description and observations. Suppose that, after repairing some components Δ' in Δ, we get new observations. Then,

$$H' = (H \cup E) \setminus F$$

with $E = \{\neg Ab(c) \mid c \in \Delta'\}$ and $F = \{Ab(c) \mid c \in \Delta'\}$ becomes a new diagnosis. Here, positive and negative explanations are necessary at the same time. Recently, Buccafurri *et al.* (1997) apply extended abduction to formalize a system repair problem in the context of model checking, which is a successful technique for the verification of concurrent programs and protocols.

In this paper, we study the computational aspect of extended abduction. We consider knowledge bases written in normal logic programs, and introduce a program transformation which produces a *transaction program*. A transaction program is a kind of disjunctive logic program that specifies addition and deletion of abductive hypotheses. Abductive explanations are computed by the fixpoint of the transaction program using bottom-up model generation procedures like (Bry, 1990; Inoue and Sakama, 1996; Aravindan and Baumgartner, 1997). The correctness of the proposed method is shown for the class of acyclic programs. In the context of databases, a transaction program provides a declarative specification of database update. In a special case, a transaction program gives a *revision program* in the sense of Marek and Truszczynski (1994).

The rest of this paper is organized as follows. In Section 2, we outline our extended abductive framework. In Section 3, we introduce a transformation of abductive logic programs into transaction programs, and present a fixpoint computation for extended abduction. Section 4 addresses comparisons with related work, and Section 5 concludes the paper.

2 ABDUCTIVE FRAMEWORK

An abductive framework considered in this paper is defined as follows.

A *normal logic program* (NLP) is a finite set of rules of the form:

$$A_0 \leftarrow A_1, \ldots, A_m, not\, A_{m+1}, \ldots, not\, A_n \qquad (1)$$

$(n \geq m \geq 0)$, where each A_i is an atom and *not* denotes *negation as failure*. The left-hand side of a rule is the *head*, and the right-hand side is the *body*. We allow a rule with an empty head, which is called an *integrity constraint*. In this paper, the empty head of any integrity constraint is represented by the special atom $\bot$. A rule with an empty body $(n = 0)$ is called a *fact*. Each fact $A \leftarrow$ is identified with the atom A.

A program (rule, atom, term) is *ground* if it contains no variable. An NLP P is *acyclic* if there is a *level mapping* from the ground atoms of P to the natural numbers, such that for any ground rule (1) from P, the level of A_0 is higher than the level of every A_i $(i = 1, \ldots, n)$ (Apt and Bezem, 1991). P is *Horn* if no rule in P contains negation as failure, that is, $m = n$ holds for every rule of the form (1) in P. P is *definite* if it is a Horn program without integrity constraints.

The semantics of NLPs is given by the *stable model semantics* (Gelfond and Lifschitz, 1988). Given any NLP P and a set M of ground atoms, the ground Horn NLP P^M is defined as follows: a rule

$$A_0 \leftarrow A_1, \ldots, A_m$$

is in P^M iff there is a ground rule of the form (1) from P such that $\{A_{m+1}, \ldots, A_n\} \cap M = \emptyset$. Note that P^M is a possibly infinite set of *not*-free ground rules. Then, P^M has the unique minimal model, i.e., the least model. If the least model of P^M is identical to M, M is called a *stable model* of P. A stable model is *consistent* if it does not contain $\bot$, that is, it satisfies every integrity constraint. An NLP is *consistent* if it has a consistent stable model. For a consistent NLP P and an atom L, P *entails* L (written $P \models L$) if L is included in every stable model of P; otherwise $P \not\models L$. Note that any acyclic program has a unique stable model (Gelfond and Lifschitz, 1988).

An *abductive logic program* (ALP) is defined as a pair $\langle P, \mathcal{A} \rangle$, where P is an NLP and $\mathcal{A}$ is a set of atoms called *abducibles*. Any instance A of an element from $\mathcal{A}$ is also called an *abducible* and is written as $A \in \mathcal{A}$. Without loss of generality, we assume that any rule (1) having an abducible in its head ($A_0 \in \mathcal{A}$) is always a fact.[1] An ALP $\langle P, \mathcal{A} \rangle$ is *acyclic* (resp. *Horn*, *ground*) if P is acyclic (resp. Horn, ground).

Definition 2.1 Let $\langle P, \mathcal{A} \rangle$ be an ALP, and G a ground atom. A pair (E, F) is an *explanation* (resp. *anti-explanation*) of an *observation* G wrt $\langle P, \mathcal{A} \rangle$ if

1. $(P \cup E) \setminus F \models G$ (resp. $(P \cup E) \setminus F \not\models G$),
2. $(P \cup E) \setminus F$ is consistent, and
3. $E \subseteq \mathcal{A}$ and $F \subseteq \mathcal{A}$.

For an (anti-)explanation (E, F), E is also called a *positive (anti-)explanation* if $F = \emptyset$; F is called a *negative (anti-)explanation* if $E = \emptyset$; and (E, F) is called a *mixed (anti-)explanation* if $E \neq \emptyset$ and $F \neq \emptyset$.

An (anti-)explanation (E, F) of G is *minimal* if for any (anti-)explanation (E', F') of G, $E' \subseteq E$ and $F' \subseteq F$ imply $E' = E$ and $F' = F$.

In this paper, minimal explanations are of particular interest, and (anti-)explanations mean minimal (anti-)explanations unless stated otherwise. Note that $E \cap F = \emptyset$ holds for any minimal (anti-)explanation (E, F).

Example 2.1 Consider the introductory program K in Section 1. Let $\langle K, \mathcal{A} \rangle$ be an ALP where $\mathcal{A} = \{\mathit{broken\text{-}wing}(x)\}$. Then, $G_1 = \mathit{flies}(\mathit{tweety})$ has the explanation $(E, F) = (\emptyset, \{\mathit{broken\text{-}wing}(\mathit{tweety})\})$, while $G_2 = \mathit{flies}(\mathit{opus})$ has the anti-explanation $(\{\mathit{broken\text{-}wing}(\mathit{opus})\}, \emptyset)$.

3 COMPUTING EXTENDED ABDUCTION

In this section, we provide a method of computing extended abduction. In Sections 3.1 and 3.2, programs are assumed to be ground, and results are extended to non-ground programs in Section 3.3.

3.1 TRANSACTION PROGRAM

We first introduce the notion of *transaction programs*, which are obtained by transforming abductive logic programs. This transformation is motivated to get rules for what must be in the knowledge base and what must be outside it for any observation to hold.

For any atom A, formulas $in(A)$ and $out(A)$ are called *T-literals* (or *literal* for short). A *transaction program* consists of *productions* of the form:

$$L \rightarrow \alpha_1 \mid \ldots \mid \alpha_k, \tag{2}$$

where L is a literal and α_i is a sequence of literals.[2] Here, the vertical line "|" is read "or" meaning a disjunction, while a sequence means a conjunction of literals. In a production of the form (2), the left-hand side L is called the *antecedent* and the right-hand side $\alpha_1 \mid \ldots \mid \alpha_k$ is called the *consequent* of the production.

Definition 3.1 Let $\langle P, \mathcal{A} \rangle$ be a ground ALP. Then, the *transaction program* τP *of* P *(relative to* $\mathcal{A}$*)* is defined as follows.

1. Let H be a non-abducible atom ($H \notin \mathcal{A}$) and
$$H \leftarrow B_1, \quad \vdots \quad H \leftarrow B_k,$$
be all rules in P with H in the head, where
$$B_i = A_1, \ldots, A_{m_i}, \mathit{not}\, C_1, \ldots, \mathit{not}\, C_{n_i}$$
($1 \leq i \leq k$, $m_i \geq 0$, $n_i \geq 0$). Then, these rules are transformed to the following productions:
$$R_{in}: \quad in(H) \rightarrow in(B_1) \mid \ldots \mid in(B_k), \tag{3}$$
$$R_{out}: \quad out(H) \rightarrow out(B_1), \quad \vdots \quad out(H) \rightarrow out(B_k), \tag{4}$$
where for $i = 1, \ldots, k$, $in(B_i)$ and $out(B_i)$ are defined as follows. If B_i is empty, $in(B_i) = \varepsilon$; otherwise $in(B_i)$ is:
$$in(A_1), \ldots, in(A_{m_i}), out(C_1), \ldots, out(C_{n_i}).$$
If B_i is empty, $out(B_i) = \mathit{false}$; else $out(B_i)$ is:
$$out(A_1) \mid \ldots \mid out(A_{m_i}) \mid in(C_1) \mid \ldots \mid in(C_{n_i}).$$
When $H = \perp$ (integrity constrains), only R_{out} is included in τP.
2. When $H \leftarrow$ is in P for $H \in \mathcal{A}$ (note in this case that there is no rule with H in the head and with a non-empty body by the assumption [footnote 1]), it is transformed to
$$R_{in}: \quad in(H) \rightarrow \varepsilon,$$
and we define that R_{out} is empty.

[1]If there is a rule $A \leftarrow \Gamma$ with an abducible A and a non-empty body Γ, then A is made a non-abducible by introducing a rule $A \leftarrow A'$ with a new abducible A'.

[2]In non-ground transaction programs, productions may contain equalities or inequalities in their antecedents or consequents (see Definition 3.4).

3. Let A be any atom that does not appear in the head of any rule in P. For every such atom A, introduce the production:

$$out(A) \rightarrow \varepsilon \quad (5)$$

to τP. Moreover, when A is a non-abducible ($A \notin \mathcal{A}$), τP also includes:

$$in(A) \rightarrow false. \quad (6)$$

Each production generated in the first step specifies abductive specifications such that to explain H, one of $B_1, \dots, B_k$ must be explained (3), while to unexplain H, every $B_1, \dots, B_k$ must be unexplained (4). The meanings of $in(B_i)$ and $out(B_i)$ are obvious, that is, introduction and removal of B_i respectively, which correspond to explaining/unexplaining B_i. When B_i is empty, there is nothing to be introduced and $in(B_i) = \varepsilon$. Here, the empty string ε is logically equivalent to *true* and $\{\varepsilon\} \cup S$ is identical to S for any set S. On the other hand, $out(B_i) = false$ means that removing a non-abducible H with the empty body is impossible. Note that, for integrity constraints, the removal of $\perp$ means satisfaction of all constraints. In the second step, when an abducible H is in P, nothing is required to add H, while no further transaction is requested to remove H. In the third step, additional productions (5) mean that the removal of A implies no requirement if P has no rule with A in the head, while productions (6) mean that the addition of A is impossible.

Example 3.1 Let $\langle P, \mathcal{A} \rangle$ be an ALP, where

$$\begin{aligned}
P:\ & p \leftarrow q, not\, a, \\
& p \leftarrow b, not\, r, \\
& q \leftarrow not\, c, \\
& r \leftarrow d, \\
& c \leftarrow, \\
& d \leftarrow, \\
\mathcal{A}:\ & a,\ b,\ c,\ d.
\end{aligned}$$

Then, τP becomes

$$\begin{aligned}
& in(p) \rightarrow in(q), out(a) \mid in(b), out(r), \\
& out(p) \rightarrow out(q) \mid in(a), \\
& out(p) \rightarrow out(b) \mid in(r), \\
& in(q) \rightarrow out(c), \\
& out(q) \rightarrow in(c), \\
& in(r) \rightarrow in(d), \\
& out(r) \rightarrow out(d), \\
& in(c) \rightarrow \varepsilon, \\
& in(d) \rightarrow \varepsilon, \\
& out(a) \rightarrow \varepsilon, \\
& out(b) \rightarrow \varepsilon.
\end{aligned}$$

3.2 FIXPOINT COMPUTATION

For computing extended abduction, we consider fixpoint computation of transaction programs. The $\mathbf{T}_{\tau P}$ *fixpoint operator* which we define next is similar to that given by Inoue and Sakama (1996) for disjunctive logic programs. The only difference between the operator below and that in (Inoue and Sakama, 1996) lies in the notion of *marking*. Since each production can also be viewed as a *rewriting rule*, once a literal L is rewritten it is marked as $L*$. In this way, it is recognized that the requirement of addition/deletion of L has been translated into requests of another atoms' addition/deletion.

Formally, for T-literals $in(A)$ and $out(A)$, formulas $in(A)*$ and $out(A)*$ are also called *T-literals* (or *literals* for short). T-literals with the "$*$" symbol are called *marked literals*, while other literals are *unmarked*. Given a transaction program τP, let $\mathcal{B}$ be the set of ground T-literals in the language of τP, i.e., the Herbrand base. A *transaction* is a subset of $\mathcal{B}$, i.e., an Herbrand interpretation. Each transaction specifies which atoms should be added or deleted. A transaction S is *coherent* if for any atom A, S does not include any pair of $\{in(A), out(A)\}$, $\{in(A)*, out(A)\}$, $\{in(A), out(A)*\}$ and $\{in(A)*, out(A)*\}$.

A transaction S *satisfies* a T-literal L (written $S \models L$), where L is either $in(A)$ or $out(A)$, if either $L \in S$ or $L* \in S$ holds. For a conjunction of literals, we also write $S \models L_1, \dots, L_n$ if $S \models L_i$ for every $i = 1, \dots, n$. A transaction S *satisfies* a production ($L \rightarrow \alpha_1 \mid \dots \mid \alpha_k$) if $S \models L$ implies $S \models \alpha_i$ for some i ($1 \leq i \leq k$).

Now, we define a mapping over sets of transactions. For this purpose, two operations *marking* ($*$) and *filtering* ($\dot{-}$) are defined as follows. For transactions I and J,

$$I * J = (I \setminus J) \cup \{L* \mid L \in I \cap J\},$$
$$I \dot{-} J = \{L \in I \mid L \notin J \text{ and } L* \notin J\}.$$

Intuitively, $I*J$ denotes that each literal in J is marked in I, and $I \dot{-} J$ means that a literal in I is removed from I if it is included in J in either marked or unmarked form. Also, for a conjunction of literals $F = L_1, \dots, L_m$, we denote the set of its conjuncts as $\{F\} = \{L_1, \dots, L_m\}$.

Definition 3.2 Let τP be a ground transaction program, $\mathbf{I}$ a set of transactions. Then the mapping $\mathbf{T}_{\tau P} : 2^{2^{\mathcal{B}}} \rightarrow 2^{2^{\mathcal{B}}}$ is defined as

$$\mathbf{T}_{\tau P}(\mathbf{I}) = \bigcup_{I \in \mathbf{I}} \{\, J \in T_{\tau P}(I) \mid J \text{ is coherent} \,\}$$

where the mapping $T_{\tau P} : 2^{\mathcal{B}} \rightarrow 2^{2^{\mathcal{B}}}$ is defined as follows. For any transaction I, $T_{\tau P}(I)$ is equal to:

$$\begin{cases} \emptyset, & \text{if } I \models L \text{ for some production } (L \rightarrow \mathit{false}) \text{ in } P; \\ \{\, (I * J) \cup (K \dot{-} I) \mid J = \{L_i \mid R_i \in \mathbf{R}\} \text{ and } K = \bigcup_{R_i \in \mathbf{R}} \{\alpha^i\}, \\ \quad \text{where } \mathbf{R} \text{ is the set of productions of the form: } R_i : (L_i \rightarrow F_1 \mid \dots \mid F_{k_i}) \\ \quad \text{in } P \text{ such that } I \models L_i \text{ and that } \{F_j\} \dot{-} I \neq \emptyset \text{ for every } j = 1, \dots, k_i, \\ \quad \text{and } \alpha^i \text{ is one of } F_1, \dots, F_{k_i} \,\}, & \text{otherwise.} \end{cases}$$

In particular, $\mathbf{T}_{\tau P}(\emptyset) = \emptyset$.

The intuitive reading of Definition 3.2 is as follows. If a transaction I does not satisfy a production with false in the consequent, then I is pruned. Else, if there is a production R_i that is not satisfied by I (i.e., I satisfies the antecedent of R_i but does not satisfy the consequent of R_i), then I is expanded by adding each single disjunct F_j in the consequent for every such R_i, so that R_i is satisfied by I. In expanding I with α^i's, each antecedent of R_i in I is marked, then only literals in α^i filtered by I are added. In the case that R_i is $(L \rightarrow \varepsilon)$, L is simply marked and nothing is added with this production.

The ordinal powers of $\mathbf{T}_{\tau P}$ are defined as follows.

$$\begin{array}{lcl} \mathbf{T}_{\tau P} \uparrow 0 & = & \{\emptyset\}, \\ \mathbf{T}_{\tau P} \uparrow n+1 & = & \mathbf{T}_{\tau P}(\mathbf{T}_{\tau P} \uparrow n), \\ \mathbf{T}_{\tau P} \uparrow \omega & = & \bigcup_{\alpha<\omega} \bigcap_{\alpha \le n < \omega} \mathbf{T}_{\tau P} \uparrow n, \end{array}$$

where n is a successor ordinal and ω is a limit ordinal. The above definition means that at the limit ordinal ω the closure retains transactions which are persistent in the preceding computation. That is, for any transaction I in $\mathbf{T}_{\tau P} \uparrow \omega$, there is an ordinal α smaller than ω such that, for every n ($\alpha \le n < \omega$), I is included in $\mathbf{T}_{\tau P} \uparrow n$. This closure definition is also used in (Inoue and Sakama, 1996) for computing minimal models of disjunctive logic programs. By the result in (Inoue and Sakama, 1996), $\mathbf{T}_{\tau P} \uparrow \omega$ becomes a fixpoint.

Example 3.2 Consider the transaction program τP in Example 3.1. Let $\tau P^{+p} = \tau P \cup \{ \rightarrow in(p) \}$. Then, the fixpoint closure of τP^{+p} becomes

$$\begin{array}{lcl} \mathbf{T}_{\tau P^{+p}} \uparrow 1 & = & \{ \{in(p)\} \}, \\ \mathbf{T}_{\tau P^{+p}} \uparrow 2 & = & \{ \{in(p)*, in(q), out(a)\}, \\ & & \ \{in(p)*, in(b), out(r)\} \}, \\ \mathbf{T}_{\tau P^{+p}} \uparrow 3 & = & \{ \{in(p)*, in(q)*, out(c), out(a)*\}, \\ & & \ \{in(p)*, in(b), out(r)*, out(d)\} \}, \\ \mathbf{T}_{\tau P^{+p}} \uparrow 4 & = & \mathbf{T}_{\tau P^{+p}} \uparrow 3. \end{array}$$

Thus, $\mathbf{T}_{\tau P^{+p}} \uparrow \omega = \mathbf{T}_{\tau P^{+p}} \uparrow 3$.

Note that the above fixpoint computation constructs minimal models in a *bottom-up* manner when productions are viewed as disjunctive clauses. *Model generation procedures* such as SATCHMO, MGTP, and Hyper-Tableaux used in (Bry, 1990; Inoue and Sakama, 1996; Aravindan and Baumgartner, 1997) are similar to this computation mechanism. They perform hyperresolution to expand Herbrand interpretations and case-splitting on non-unit hyperresolvents. When a program is finite, function-free and range-restricted,[3] closure computation by these procedures stops in a finite step. For a ground transaction program, these conditions are always satisfied.

We are ready to compute extended abduction from the fixpoint of a transaction program. T-literals $in(A)/out(A)$ with $A \in \mathcal{A}$ are called *abducible literals.* For a transaction S, the set of *unmarked abducible literals* in S is denoted as S°. For a set of transactions $\mathbf{I} \subseteq 2^{\mathcal{B}}$, let

$$min^\circ(\mathbf{I}) = \{ S^\circ \mid S \in \mathbf{I} \text{ and there is no } S' \in \mathbf{I} \text{ such that } S'^\circ \subset S^\circ \}.$$

Let us denote a transaction program with an observation G to be (un)explained as:

$$\begin{array}{lcl} \tau P^{+G} & = & \tau P \cup \{ \rightarrow in(G), \ \rightarrow out(\bot) \}, \\ \tau P^{-G} & = & \tau P \cup \{ \rightarrow out(G), \ \rightarrow out(\bot) \}. \end{array}$$

Here, $out(\bot)$ means that all integrity constraints should be satisfied. When there is no integrity constraint in P, $(\rightarrow out(\bot))$ is not added to τP^{+G} and τP^{-G}. Also, we write

$$\begin{array}{lcl} IN(S) & = & \{A \mid in(A) \in S\}, \\ OUT(S) & = & \{A \mid out(A) \in S\}. \end{array}$$

Definition 3.3 For an acyclic ALP $\langle P, \mathcal{A} \rangle$ and a ground observation G, *abduction steps in P* are inductively defined as follows.

1. G is *explainable by* $(\emptyset, \emptyset)$ *at 0-step* if $G \in P$.
   G is *explainable by* $(\{G\}, \emptyset)$ *at 0-step* if $G \in \mathcal{A}$.

2. G is *unexplainable by* $(\emptyset, \emptyset)$ *at 0-step* if P has no ground rule from P with G in the head.
   G is *unexplainable by* $(\emptyset, \{G\})$ *at 0-step* if $G \in P$ and $G \in \mathcal{A}$.

3. G is *explainable by* (E, F) *at* $(s+1)$*-step* if there is a ground rule:

   $$G \leftarrow A_1, \dots, A_m, not\, A_{m+1}, \dots, not\, A_n$$

   from P such that (i) each A_i ($i = 1, \dots, m$) is explainable by (E_i, F_i) at l_i-step ($l_i \le s$) and each A_j ($j = m, \dots, n$) is unexplainable by (E_j, F_j) at l_j-step ($l_j \le s$), (ii) $E = \bigcup_{i=1}^m E_i$, $F = \bigcup_{j=1}^n F_j$, $E \cap F = \emptyset$, and (iii) $max(l_1, \dots, l_m, l_{m+1}, \dots, l_n) = s$.

[3]A program P is *range-restricted* if for every rule in P, any variable appearing in the head has an occurrence in a positive atom in the body.

4. G is *unexplainable by* (E, F) *at* $(s+1)$*-step* if for every ground rule r_k of the form:

$$G \leftarrow A_1, \ldots, A_{m_k}, \, not\, A_{m_k+1}, \ldots, not\, A_{n_k}$$

from P, (i) there is an atom A_{i_k} $(1 \leq i_k \leq n_k)$ such that A_{i_k} is unexplainable (when $1 \leq i_k \leq m_k$) or explainable (when $m_k + 1 \leq i_k \leq n_k$) by (E_k, F_k) at l_k-step $(l_k \leq s)$, (ii) $E = \bigcup_{r_k} E_k$, $F = \bigcup_{r_k} F_k$, $E \cap F = \emptyset$, and (iii) $max_{r_k} l_k = s$.

In particular, we say that G is *(un)provable in* P at s-step for some $s < \omega$ if G is (un)explainable by $(\emptyset, \emptyset)$ in P at s-step. Note that this definition of provability (resp. unprovability) of G in P characterizes a positive (resp. negative) conclusion about G in P under the *well-founded semantics* (van Gelder *et al.*, 1991). Namely, G is true/false under the well-founded model of P iff G is provable/unprovable in P. Since the well-founded model and the stable model coincide in acyclic NLPs (van Gelder *et al.*, 1991), it holds that: for an acyclic NLP P, $P \models G$ (under the stable model semantics) iff G is provable in P at s-step.

Lemma 3.1 *Let* $\langle P, \mathcal{A} \rangle$ *be an acyclic ALP,* G *a ground observation, and* $E, F \in \mathcal{A}$*. Suppose that* $(P \cup E) \setminus F$ *is consistent.*

(i) *If* G *is (un)explainable by* (E, F) *in* P *at* s*-step for some* $s < \omega$*, then* (E, F) *is an (anti-)explanation of* G *wrt* $\langle P, \mathcal{A} \rangle$*.*

(ii) *If* (E, F) *is a minimal (anti-)explanation of* G *wrt* $\langle P, \mathcal{A} \rangle$*, then* G *is (un)explainable by* (E, F) *in* P *at* s*-step for some* $s < \omega$*.*

Proof: The proof of (i) is obvious by induction on the abduction step in P.

(ii) Let (E, F) be a minimal explanation of G wrt $\langle P, \mathcal{A} \rangle$. Then, $(P \cup E) \setminus F \models G$. By the discussion above, G is provable in $(P \cup E) \setminus F$ at s-step for some $s < \omega$. In other words, G is explainable by $(\emptyset, \emptyset)$ in $(P \cup E) \setminus F$ at s-step. Then, G is explainable by $(\emptyset, F)$ in $P \cup E$ at s-step, because (E, F) is a minimal explanation of G and hence no $F' \subset F$ satisfies that G is explainable by $(\emptyset, F')$ in $P \cup E$. Then, G is explainable by (E, F) in P at s-step, because (E, F) is a minimal explanation of G and hence no $E' \subset E$ satisfies that G is explainable by (E', F) in P.

The proof for a minimal anti-explanation can be shown in a similar way. □

Theorem 3.2 *Let* $\langle P, \mathcal{A} \rangle$ *be a ground acyclic ALP, and* G *a ground observation. Then,*

(i) (E, F) *is a minimal explanation of* G *wrt* $\langle P, \mathcal{A} \rangle$ *iff there is a transaction* $S \in min^\circ(\mathbf{T}_{\tau P+G} \uparrow \omega)$ *such that* $E = IN(S)$ *and* $F = OUT(S)$*.*

(ii) (E, F) *is a minimal anti-explanation of* G *wrt* $\langle P, \mathcal{A} \rangle$ *iff there is a transaction* $S \in min^\circ(\mathbf{T}_{\tau P-G} \uparrow \omega)$ *such that* $E = IN(S)$ *and* $F = OUT(S)$*.*

Proof: We prove that G is (un)explainable by (E, F) at s-step for some $s < \omega$ iff there is a transaction S in $min^\circ(\mathbf{T}_{\tau P+G} \uparrow \omega)$ (or $min^\circ(\mathbf{T}_{\tau P-G} \uparrow \omega)$) such that $E = IN(S)$ and $F = OUT(S)$. This is proved by induction on the abduction step s. Then, the results (i) and (ii) follow by Lemma 3.1. Suppose first that $s = 0$.

(i) If G is explainable by (E, F) at 0-step, (E, F) is either $(\emptyset, \emptyset)$ or $(\{G\}, \emptyset)$. In the former case, $G \in P$ holds, and then $(in(G) \rightarrow \epsilon)$ is in τP^{+G}. Hence, $min^\circ(\mathbf{T}_{\tau P+G} \uparrow \omega) = \{\emptyset\}$, $E = IN(\emptyset)$ and $F = OUT(\emptyset)$. In the latter case, $G \in \mathcal{A}$ holds, and then $\{in(G)\} \in min^\circ(\mathbf{T}_{\tau P+G} \uparrow \omega)$, $E = IN(\{in(G)\})$ and $F = OUT(\{in(G)\})$.

(ii) If G is unexplainable by (E, F) at 0-step, (E, F) is either $(\emptyset, \emptyset)$ of $(\emptyset, \{G\})$. In the former case, P has no rule with G in the head. Then, $(out(G) \rightarrow \epsilon)$ is in τP^{-G}, hence $min^\circ(\mathbf{T}_{\tau P-G} \uparrow \omega) = \{\emptyset\}$, $E = IN(\emptyset)$ and $F = OUT(\emptyset)$. In the latter case, $G \in P \cap \mathcal{A}$ holds. Then, $\{out(G)\} \in min^\circ(\mathbf{T}_{\tau P-G} \uparrow \omega)$, $E = IN(\{out(G)\})$ and $F = OUT(\{out(G)\})$.

Next suppose that the results hold for every step $l \leq s$.

(i) Suppose that G is explainable by (E, F) at $(s+1)$-step, where (E, F) is a minimal explanation of G. Since P is acyclic, there is a ground rule

$$G \leftarrow A_1, \ldots, A_m, \, not\, A_{m+1}, \ldots, not\, A_n$$

in P such that (i) each A_i $(i = 1, \ldots, m)$ is explainable by (E_i, F_i) at l_i-step $(l_i \leq s)$ and each A_j $(j = m, \ldots, n)$ is unexplainable by (E_j, F_j) at l_j-step $(l_j \leq s)$, (ii) $E = \bigcup_{i=1}^m E_i$, $F = \bigcup_{j=1}^n F_j$, and $E \cap F = \emptyset$. By the induction hypothesis, each (E_i, F_i) is a minimal explanation of A_i iff $S_i \in min^\circ(\mathbf{T}_{\tau P+A_i} \uparrow \omega)$ such that $E_i = IN(S_i)$ and $F_i = OUT(S_i)$. Likewise, each (E_j, F_j) is a minimal anti-explanation of A_j iff $S_j \in min^\circ(\mathbf{T}_{\tau P-A_j} \uparrow \omega)$ such that $E_j = IN(S_j)$ and $F_j = OUT(S_j)$.

On the other hand, there is a production

$$in(G) \rightarrow in(B_1) \mid \ldots \mid in(B_l)$$

in τP such that for some $1 \leq k \leq l$, $in(B_k)$ is equal to

$$in(A_1), \ldots, in(A_m), out(A_{m+1}), \ldots, out(A_n).$$

Let $S = \{in(G)*\} \cup \bigcup_{i=1}^n S_i$. Here, $E \cap F = \emptyset$ implies that S is coherent. Then, $S \in \mathbf{T}_{\tau P+G} \uparrow \omega$. Also, the minimality of each (E_i, F_i) implies the minimality of S°. Therefore, $S^\circ \in min^\circ(\mathbf{T}_{\tau P+G} \uparrow \omega)$ and $E = IN(S^\circ)$ and $F = OUT(S^\circ)$.

(ii) Suppose that G is unexplainable by (E, F) at $(s+$

1)-step, where (E, F) is a minimal anti-explanation of G. Then, for every ground rule of the form

$$r_k : (G \leftarrow A_1, \ldots, A_{m_k}, \mathit{not}\, A_{m_k+1}, \ldots, \mathit{not}\, A_{n_k})$$

in P, there is an atom A_{i_k} $(1 \leq i_k \leq n_k)$ such that A_{i_k} is unexplainable (when $1 \leq i_k \leq m_k$) or explainable (when $m_k + 1 \leq i_k \leq n_k$) by (E_k, F_k) at l_k-step $(l_k \leq s)$, (ii) $E = \bigcup_{r_k} E_k$, $F = \bigcup_{r_k} F_k$, and $E \cap F = \emptyset$. By the induction hypothesis, each (E_k, F_k) is a minimal anti-explanation of A_{i_k} $(1 \leq i_k \leq m_k)$ iff $S_k \in min^{\circ}(\mathbf{T}_{\tau P^{-A_{i_k}}} \uparrow \omega)$ such that $E_k = IN(S_k)$ and $F_k = OUT(S_k)$. Likewise, each (E_k, F_k) is a minimal explanation of A_{i_k} $(m_k + 1 \leq i_k \leq n_k)$ iff $S_k \in min^{\circ}(\mathbf{T}_{\tau P^{+A_{i_k}}} \uparrow \omega)$ such that $E_k = IN(S_k)$ and $F_k = OUT(S_k)$.

On the other hand, there are productions

$$(out(G) \rightarrow out(B_1)),\ \ldots,\ (out(G) \rightarrow out(B_u))$$

in τP, where each $out(B_i)$ is a disjunction of literals. Then, these productions can be rewritten to one production with $out(G)$ in the antecedent:

$$out(G) \rightarrow out(B'_1) \mid \ldots \mid out(B'_v),$$

where each $out(B'_i)$ is now a conjunction of literals formed by collecting one disjunct from every disjunction $out(B_k)$ for $k = 1, \ldots, u$. Then, there is an $out(B'_h)$ which contains either $S_k \in min^{\circ}(\mathbf{T}_{\tau P^{-A_{i_k}}} \uparrow \omega)$ $(1 \leq i \leq m_k)$ or $S_k \in min^{\circ}(\mathbf{T}_{\tau P^{+A_{i_k}}} \uparrow \omega)$ $(m_k+1 \leq j \leq n_k)$ for every r_k. Let $S = \{out(G)*\} \cup \{out(B'_h)\}$. Here, $E \cap F = (\bigcup_{r_k} E_k) \cap (\bigcup_{r_k} F_k) = \emptyset$ implies that S is coherent. Then, by the fixpoint construction, $S \in \mathbf{T}_{\tau P^{-G}} \uparrow \omega$. Also, the minimality of each (E_k, F_k) implies the minimality of S°. Hence, $S^{\circ} \in min^{\circ}(\mathbf{T}_{\tau P^{-G}} \uparrow \omega)$. By $E = \bigcup_{r_k} IN(S_k)$ and $F = \bigcup_{r_k} OUT(S_k)$, $E = IN(S^{\circ})$ and $F = OUT(S^{\circ})$.

To show the converse direction, put $E = IN(S)$ and $F = OUT(S)$ for $S \in min^{\circ}(\mathbf{T}_{\tau P^{+G}} \uparrow \omega)$ (resp. $S \in min^{\circ}(\mathbf{T}_{\tau P^{-G}} \uparrow \omega)$). Then, we can show that G is explainable (resp. unexplainable) by (E, F) at s-step by following the above proofs backward. □

Corollary 3.3 *A ground observation G has no explanation (resp. anti-explanation) in a ground acyclic ALP $\langle P, \mathcal{A} \rangle$ if $\mathbf{T}_{\tau P^{+G}} \uparrow \omega = \emptyset$ (resp. $\mathbf{T}_{\tau P^{-G}} \uparrow \omega = \emptyset$).*

Example 3.3 In Example 3.2, $min^{\circ}(\mathbf{T}_{\tau P^{+p}} \uparrow \omega) = \{\{out(c)\}, \{in(b), out(d)\}\}$. Then, $G = p$ has two minimal explanations $(E, F) = (\emptyset, \{c\})$, $(\{b\}, \{d\})$.

When a program contains cycles, Theorem 3.2 does not hold in general.

Example 3.4 Let P be the program

$$p \leftarrow q,$$
$$q \leftarrow p,$$

and $\mathcal{A} = \emptyset$. Then, p has no explanation. However,

$$in(p) \rightarrow in(q),$$
$$in(q) \rightarrow in(p)$$

are in τP, then $\mathbf{T}_{\tau P^{+p}} \uparrow \omega = \{\{in(p)*, in(q)\}\}$ and $min^{\circ}(\mathbf{T}_{\tau P^{+p}} \uparrow \omega) = \{\emptyset\}$, which does not imply the absence of explanation.

However, the acyclicity is not a necessary condition for Theorem 3.2.

Example 3.5 Let P be the program

$$g \leftarrow p,$$
$$p \leftarrow \mathit{not}\, q,$$
$$q \leftarrow q,$$

and $\mathcal{A} = \emptyset$. Then, g has the explanation $(E, F) = (\emptyset, \emptyset)$. In this case, τP includes

$$in(g) \rightarrow in(p),$$
$$in(p) \rightarrow out(q),$$
$$out(q) \rightarrow out(q).$$

Then, $\mathbf{T}_{\tau P^{+g}} \uparrow \omega = \{\{in(g)*, in(p)*, out(q)\}\}$, and $min^{\circ}(\mathbf{T}_{\tau P^{+g}} \uparrow \omega) = \{\emptyset\}$, which corresponds to the explanation.

3.3 NON-GROUND CASE

We extend results in previous subsections to non-ground programs. In this section, we consider a non-ground ALP $\langle P, \mathcal{A} \rangle$ in which P is a *covered* NLP and observations are ground. Here, $\langle P, \mathcal{A} \rangle$ is called a *covered* ALP if P is *covered*, that is, for every rule in P, all variables in the body appear in the head (Subrahmanian, 1987).[4] With this restriction, transactions in any bottom-up computation are guaranteed to be always ground. We also assume that for each predicate p, every atom with p is either abducible or non-abducible, and call such a predicate an *abducible predicate* in the former case. While we might consider a more general case using theories of predicate completion in logic programming, such an extension is too technical and is beyond the scope of this paper.

In the presence of variables in a program, we use *Clark's completion* (Clark, 1978) for producing transaction programs. Recall that Clark's completion is defined as follows. Any rule

$$p(t_1, \ldots, t_n) \leftarrow B$$

in P, where $t_1, \ldots, t_n$ are terms, is written in the form

$$p(x_1, \ldots, x_n) \leftarrow (\exists \mathbf{y})\,(x_1 = t_1), \ldots, (x_n = t_n),\ B,$$

where $x_1, \ldots, x_n$ are new variables and $\mathbf{y}$ are the variables in the original rule. Note here that when P is

[4]A covered program is also called a ***constrained*** program in the literature, e.g., (Lavrač and Džeroski, 1994).

covered and each t_i is ground, $(\exists y)$ need not be considered. If there are k rules with the predicate p in the head, they are written in the form

$$\begin{aligned} &p(x_1,\ldots,x_n) \leftarrow E_1, B_1,\\ &\quad\vdots \qquad\qquad\qquad\qquad\qquad\qquad (7)\\ &p(x_1,\ldots,x_n) \leftarrow E_k, B_k, \end{aligned}$$

where E_i's are equalities and B_i's are original bodies. Then, the definition for p is

$$p(x_1,\ldots,x_n) \leftrightarrow E_1, B_1 \mid \ldots \mid E_k, B_k.$$

Now, given an ALP $\langle P, \mathcal{A}\rangle$, its transaction program is defined like Definition 3.1 based on Clark's completion.

Definition 3.4 Let $\langle P, \mathcal{A}\rangle$ be a covered ALP. The *transaction program τP of P (relative to $\mathcal{A}$)* is defined as follows.

1. Let p be a non-abducible predicate and $k > 0$ for rules (7). Then, for $H = p(x_1,\ldots,x_n)$, rules (7) are transformed to productions:

   $$R_{in}: \; in(H) \rightarrow E_1, in(B_1) \mid \ldots \mid E_k, in(B_k), \quad (8)$$

   $$\begin{aligned} R_{out}: \quad & out(H),\, E_1 \rightarrow out(B_1),\\ & \quad\vdots \qquad\qquad\qquad\qquad\qquad (9)\\ & out(H),\, E_k \rightarrow out(B_k),\\ & out(H),\, \neg E_1,\ldots,\neg E_k \rightarrow \varepsilon, \qquad (10) \end{aligned}$$

   where for $i = 1,\ldots,k$, $in(B_i)$ and $out(B_i)$ are defined in the same way as Definition 3.1. In particular, if B_i is empty, then $in(B_i) = \varepsilon$ and $out(B_i) = false$. Also, $\neg E = (x_1 \neq t_1) \vee \ldots \vee (x_n \neq t_n)$ for $E = (x_1 = t_1),\ldots,(x_n = t_n)$.

   When $H = \perp$ (integrity constrains), there are no E_i's and only (9) in R_{out} is included in τP.

2. Let p be an abducible predicate and $k > 0$ for rules (7). In this case, every rule in (7) is in the form $p(t_1,\ldots,t_n) \leftarrow$ by the assumption.[5] Then, for $H = p(x_1,\ldots,x_n)$, rules (7) are transformed to

   $$\begin{aligned} R_{in}: \quad & in(H),\, E_1 \rightarrow \varepsilon,\\ & \quad\vdots \qquad\qquad\qquad\qquad\qquad (11)\\ & in(H),\, E_k \rightarrow \varepsilon,\\ R_{out}: \quad & out(H),\, \neg E_1,\ldots,\neg E_k \rightarrow \varepsilon. \qquad (12) \end{aligned}$$

3. Let p be any predicate whose definition is empty, i.e., $k = 0$ for rules (7). For every such predicate p, introduce the production:

   $$out(p(x_1,\ldots,x_n)) \rightarrow \varepsilon \qquad (13)$$

   to τP. Moreover, when p is a non-abducible predicate, τP also includes:

   $$in(p(x_1,\ldots,x_n)) \rightarrow false. \qquad (14)$$

[5] Recall that any rule with an abducible in the head is always a fact (see Section 2).

An essential difference from the ground case is that transaction programs now contain equalities E_i's and inequalities $\neg E_i$'s. In a covered program with a ground observation, we can regard each equality $x = t$ or inequality $x \neq t$ just as a literal to be evaluated for comparison of terms: $x = t$ is *true* ($x \neq t$ is *false*) if the terms currently instantiating x and t are literally identical or they are unifiable; otherwise $x = t$ is *false* ($x \neq t$ is *true*). (In)equalities in antecedents of productions are thus evaluated for this checking after T-literals in antecedents are instantiated. Then, if x and t are unifiable in an equality $x = t$ in the antecedent of some production (9), every occurrence of t in T-literals in the consequent of the production is substituted with the term currently instantiating x. On the other hand, if x and t are unifiable in $x = t$ in the consequent of (8), every occurrence of t in the same disjunct of the consequent is substituted with the term currently instantiating x. Note that when an NLP P is ground, Definition 3.4 reduces to Definition 3.1.

The mapping $T_{\tau P}$ in Definition 3.2 is also slightly extended as follows. Variables in each production are instantiated with a substitution σ such that $in(H\sigma)$ or $out(H\sigma)$ in the antecedent is contained in a transaction I. Marking is performed upon an instantiated literal in the antecedent of a production. Here, in obtaining a substitution σ in the $T_{\tau P}$ operator, it is sufficient to consider *matching* instead of full unification if a given NLP is covered and an observation is ground. Starting from a ground observation $in(G)$ or $out(G)$, the fixpoint operator is repeatedly applied as long as some transaction is changed. In this case, every transaction I constructed in fixpoint computation contains only ground literals. Model generation procedures introduced in Section 3.2 can be used in this case too, since τP is range-restricted whenever P is constrained.

With this setting, fixpoint closures are defined in the same manner as in the preceding section, and $\mathbf{T}_{\tau P} \uparrow \omega$ reaches the fixpoint. Then, results of Theorem 3.2 are directly extended to computing (anti-)explanations from non-ground acyclic programs.

Theorem 3.4 *Let $\langle P, \mathcal{A}\rangle$ be a covered acyclic ALP, and G a ground observation. Then,*

(i) *(E, F) is a minimal explanation of G wrt $\langle P, \mathcal{A}\rangle$ iff there is a transaction $S \in min^{\circ}(\mathbf{T}_{\tau P+G} \uparrow \omega)$ such that $E = IN(S)$ and $F = OUT(S)$.*

(ii) *(E, F) is a minimal anti-explanation of G wrt $\langle P, \mathcal{A}\rangle$ iff there is a transaction $S \in min^{\circ}(\mathbf{T}_{\tau P-G} \uparrow \omega)$ such that $E = IN(S)$ and $F = OUT(S)$.*

Proof: The proof of Theorem 3.2 can be lifted to the non-ground case. □

Example 3.6 Let $\langle P, \mathcal{A} \rangle$ be an ALP, where

$$\begin{aligned} P: \quad & g(x) \leftarrow p(x),\ not\, q(x), \\ & q(a) \leftarrow, \\ \mathcal{A}: \quad & p(x),\ q(x). \end{aligned}$$

Then, τP includes

$$\begin{aligned} & in(g(y)) \rightarrow (y = x),\ in(p(x)),\ out(q(x)), \\ & out(g(y)),\ (y = x) \rightarrow out(p(x)) \mid in(q(x)), \\ & out(p(x)) \rightarrow \varepsilon, \\ & in(q(x)),\ (x = a) \rightarrow \varepsilon, \\ & out(q(x)),\ (x \neq a) \rightarrow \varepsilon. \end{aligned}$$

Let $G1 = g(a)$ be an observation. By $min^{\circ}(\mathbf{T}_{\tau P+G1} \uparrow \omega) = \{\{in(p(a)), out(q(a))\}\}$, the minimal explanation of $G1$ is $(\{p(a)\}, \{q(a)\})$.

Next, let $G2 = g(b)$ be another (negative) observation. Then, $\mathbf{T}_{\tau P-G2} \uparrow \omega$ includes $\{out(g(b))*, out(p(b))*\}$ and $\{out(g(b))*, in(q(b))\}$, so $min^{\circ}(\mathbf{T}_{\tau P-G2} \uparrow \omega) = \{\emptyset\}$. Hence, $(\emptyset, \emptyset)$ is the minimal anti-explanation of $G2$.

4 DISCUSSION

4.1 ABDUCTIVE PROCEDURE

The idea of computing abduction through transaction programs is close to that of computing abduction through *completion* (Console *et al.*, 1991; Konolige, 1992). Our approach is different from them in the following points. First, our procedure is intended to compute positive, negative and mixed (anti-)explanations in extended abduction, while (Console *et al.*, 1991; Konolige, 1992) compute positive explanations only. In fact, productions R_{out} are newly introduced in our transformation. Second, we generate a transaction program which declaratively specifies an abductive procedure, while no such program is generated in those work. In (Console *et al.*, 1991), soundness and completeness are guaranteed for hierarchical NLPs. In our procedure, they are guaranteed for acyclic NLPs.

A fixpoint semantics of abductive logic programs is also introduced in (Inoue and Sakama, 1996), but it computes traditional abduction and does not compute the extended one. However, we have shown in this paper that a similar fixpoint operator can be used to compute extended abduction. In fact, a transaction program is a kind of disjunctive programs and fixpoint computation can be realized by model generation procedures used in (Bry, 1990; Inoue and Sakama, 1996; Aravindan and Baumgartner, 1997).

4.2 DATABASE UPDATE

It is known that abduction is used for *database update*. In deductive databases, an update request on an intensional fact is often translated into update on extensional facts. Such database update is called *view update*. More precisely, the view update problem is presented as follows. A *deductive database* (or *database* for short) is defined as a function-free consistent NLP. Let D be a database and $\mathcal{E}$ a set of ground facts in the language of D called *extensional facts*. Then, an *insertion* of a ground fact A into D (resp. *deletion* of A from D) is accomplished by an *updated database*

$$D' = (D \cup E) \setminus F \ \text{ with } \ E, F \subseteq \mathcal{E}$$

satisfying the conditions:

1. $D' \models A$ (resp. $D \not\models A$) where D' is consistent,
2. there is no database D'' satisfying the condition 1 and $D \sim D'' \subset D \sim D'$, where $D_1 \sim D_2 = (D_1 \setminus D_2) \cup (D_2 \setminus D_1)$.

The first condition says that an updated database accomplishes an insertion/deletion, while the second condition presents that the update *minimally changes* extensional facts stored in D.

Relating to abduction, an update request corresponds to an observation and extensional facts correspond to abducibles. Then, view update can be characterized by extended abduction.

Lemma 4.1 (Inoue and Sakama, 1995) *Given a database D and a ground atom A, an updated database D' accomplishes an insertion/deletion of A into/from D iff there is a minimal explanation/anti-explanation (E, F) of A in D.*

Using this lemma, the next results are obtained by Theorem 3.2.

Theorem 4.2 *Let D be a covered acyclic database, and A a ground atom.*

(i) *An insertion of A into D is achieved by an update (E, F) iff there is a transaction $S \in min^{\circ}(\mathbf{T}_{\tau D+A} \uparrow \omega)$ such that $E = IN(S)$ and $F = OUT(S)$.*

(ii) *A deletion of A from D is achieved by an update (E, F) iff there is a transaction $S \in min^{\circ}(\mathbf{T}_{\tau D-A} \uparrow \omega)$ such that $E = IN(S)$ and $F = OUT(S)$.*

Thus, we can compute database update via fixpoint computation of a transaction program.

Kakas and Mancarella (1990) characterize database update through abduction, and use a top-down abductive procedure for computing view update. In (Bry, 1990), abduction is translated into a disjunctive program and database update is realized by bottom-up computation on a meta-program specifying an update procedure. These procedures are based on traditional abduction and do not produce transaction programs.

In database update, Rossi and Naqvi (1989) use Clark's completion to compute update in Horn databases. Given an update request, it is transformed to non-Horn formulas obtained by the only-if parts of database clauses. The resulting formulas represent update on the original database. Such formulas are computed using an SLD-like top-down procedure, while they do not generate transaction programs.

In (Grant *et al.*, 1993), view update is translated into a set of disjunctive facts. This translation is based on expansion of an SLD-tree, and update is achieved by inserting/deleting these disjunctive facts to/from a database. In (Fernández *et al.*, 1996), update is achieved through construction of minimal models that satisfy an update request. In these work, databases are treated as ground programs possibly containing disjunctive facts, but transaction programs are not produced. Our transaction program specifies update by disjunction but does not introduce disjunctions to a database.

A transaction program is also viewed as a database update language which declaratively specifies dynamic changes in databases. Our program transformation provides an *automatic generation of update specification* from the original program. The generated transaction program is a kind of disjunctive program, and non-determinism in database update is naturally expressed using disjunctive productions. In this sense, transaction programs provide a new application of disjunctive logic programming. Note that several database update languages exist (Abiteboul, 1988), but few consider disjunctions in the language.

4.3 VIEW DELETION FROM DEFINITE DATABASE

Recently, Aravindan and Baumgartner (1997) proposed a transformation of ground definite programs into disjunctive programs to compute view deletion. For definite programs, R_{out} in our transaction programs is similar to their transformation, and can be used to realize view deletion in computing anti-explanations. Like (Aravindan and Baumgartner, 1997), the correctness of our fixpoint computation is guaranteed also for *non-acyclic* Horn programs.

Theorem 4.3 *Let $\langle P, \mathcal{A} \rangle$ be a ground Horn ALP, and G a ground observation. Then, F is a minimal negative anti-explanation of G wrt $\langle P, \mathcal{A} \rangle$ iff there is a transaction $S \in min^{\circ}(\mathbf{T}_{\tau P-G} \uparrow \omega)$ such that $F = OUT(S)$.*

Proof: By Lemma 4.1, a deletion of G is accomplished iff there is a minimal anti-explanation (E, F) of G wrt $\langle P, \mathcal{A} \rangle$. Since P is Horn, negation as failure does not appear and hence $E = \emptyset$, that is, F is a negative anti-explanation of G. Moreover, R_{out} in Definition 3.1 consists of productions containing *out*-literals only. Since an update request is deletion of G, only *out*-literals are concerned and R_{in} can be ignored in the transaction program τP.

Now, let S be any transaction in $\mathbf{T}_{\tau P-G} \uparrow \omega$, and $F = OUT(S^{\circ})$. Also, let $out(L)$ be any unmarked literal in S. There are two cases for L:

1. *L is an abducible and $(L \leftarrow\ ) \in P$.*
   In this case, L actually has to be removed from P.
2. *L is not an abducible.*
   In this case, L has a non-empty definition in P, because otherwise, $(out(L) \rightarrow false)$ is in τP and S is not included in the fixpoint. In this case, a production of the form:
   $$out(L) \rightarrow out(A_1) \mid \ldots \mid out(A_m)$$
   is in τP, and for some A_i $(1 \leq i \leq m)$, either $out(A_i)$ or $out(A_i)*$ is included in S. When $out(A_i) \in S$, L's removal is accomplished by the deletion of A_i. When $out(A_i)* \in S$, L's removal has already been translated into either ε when A_i has no definition, or otherwise a request of deletion of some other literals.

In the second case above, we can consider again the above two cases for A_i. If A_i is an abducible (i.e., the first case above), L's removal is successfully accomplished. Otherwise, by recursively analyzing the second case above, the chain eventually terminates as L, $L_1 (= A_i)$, ..., L_k, where L_k is either an abducible or identical to L, since the number of rules in P is finite. If L_k is an abducible, we can utilize the definition of abduction steps in Definition 3.3, so that L is unexplainable at k-step and Theorem 3.2 can be applied as well. Else if $L_k = L$, the program P is not acyclic. In this case, L_k cannot be entailed from P, and thus L_k can be unexplained and so can be L. Therefore, $out(L)$ for a non-abducible L can be removed from S by taking S°, and hence $F = OUT(S^{\circ})$ is a negative anti-explanation of G. The minimality of anti-explanations is also guaranteed by taking the operation min° to the fixpoint.

The proof of the converse direction can also be shown based on a similar argument as above. □

As (Aravindan and Baumgartner, 1997) pointed out, a general *contraction* algorithm including view deletion for definite programs requires computation of (i) all minimal positive explanations of the sentence to be contracted wrt the background knowledge without abducible facts, and then (ii) a (minimal) *hitting set* (Reiter, 1987) for these explanations, which corresponds to a (minimal) negative anti-explanation. Both Aravindan and Baumgartner's method and ours directly compute such hitting sets without computing positive explanations explicitly. In contrast to (Aravindan and

Baumgartner, 1997), our transaction programs can be applied to programs with negation as failure, and can also be used to compute positive (negative, and mixed) (anti-)explanations in extended abduction.

4.4 REVISION PROGRAM

Marek and Truszczynski (1994) define a language to specify revision in knowledge bases. The meaning of their *revision programs* is similar to the stable model semantics for logic programming. They define a *knowledge base* (KB) as a set of propositional atoms. A *(disjunctive) revision program* Π consists of *revision rules* of the form:

$$in(p_1) \mid \ldots \mid in(p_k) \mid out(r_1) \mid \ldots \mid out(r_l) \\ \leftarrow in(q_1), \ldots, in(q_m), out(s_1), \ldots, out(s_n), \quad (15)$$

where each of p_i, r_i, q_i, s_i is a propositional atom. Each revision rule in Π can be regarded as a clause in propositional logic. Then, for each minimal model M of Π, a pair (I, O) where $I = \{a \mid in(a) \in M\}$ and $O = \{a \mid out(a) \in M\}$ is called a *necessary change* for Π. Let D_I (called an *initial KB*) and D_R be two KBs. The revision program $\Pi_{D_R}|D_I$ is obtained by:

1. deleting from Π every rule of the form (15) such that $q_i \notin D_R$ for some $1 \leq i \leq m$ or $s_j \in D_R$ for some $1 \leq j \leq n$, and
2. deleting each $in(a)$ such that $a \in D_I$ and each $out(a)$ such that $a \notin D_I$ from the body of every remaining revision rule.

Let (I, O) be a necessary change for $\Pi_{D_R}|D_I$. Then, D_R is called a Π*-justified revision* of D_I if $I \cap O = \emptyset$ and $D_R = (D_I \cup I) \setminus O$. It has been shown that a Π-justified revision always performs a minimal change that is justified by the use of revision rules.

Although the direction of arrows of revision rules ($\leftarrow$) and that of productions ($\rightarrow$) are opposite, they look very similar in their syntax. In fact, a transaction program is a *context-free* revision program, i.e., a disjunctive revision program in which every revision rule has only one literal in the body. While a production of the form (2) may have a disjunctive normal form formula in the consequent, it can be equivalently written in productions with a disjunction of literals in their consequents (Inoue and Sakama, 1996), and hence corresponds to multiple revision rules.

There are several differences between the framework of (Marek and Truszczyński, 1994) and ours. First, in revision programming, a KB contains no rules except facts, and there are no notion of abducibles. Instead, a necessary change represents a whole change of atoms included in the KB, and every atom in the language can be both inserted and deleted. Second, we used the notion of marking to represent literals whose transactions have been translated into another atoms' requests, but this information should also be ignored. This is because in revision programming every atom is subject to change and view update is not considered.

Taking these differences into account, we can utilize transaction programs in revision programming. As in Section 4.2, a typical application of revision programming is database update. Recall that a database is specified by an NLP P, which is divided into the intensional part (IDB) P_r containing the rules with nonempty bodies in P, and the extensional part (EDB) containing the facts in P. An initial KB D_I is then set to a stable model of P, which includes all facts in EDB. A revision program Π is constructed from P_r and an update request by transforming them to the transaction program $\tau P_r{}^{+G}$ or $\tau P_r{}^{-G}$. Then, a revised KB D_R is computed through the transaction program. Notice that D_R still satisfies IDB of P and minimally changes EDB.

Example 4.1 Let P be the NLP:

$$p(x) \leftarrow s(x),\ not\, q(x), \\ q(a) \leftarrow, \\ s(a) \leftarrow, \\ s(b) \leftarrow .$$

Here, the first rule is transformed to the productions:

$$R: \quad in(p(y)) \rightarrow (y = x), in(s(x)), out(q(x)), \\ out(p(y)), (y = x) \rightarrow out(s(x)) \mid in(q(x)).$$

Set the initial KB to the stable model of P, i.e.,

$$D_I = \{s(a), s(b), q(a), p(b)\}.$$

Let us consider insertion of $G = p(a)$, and put

$$\Pi = R \cup \{\ \rightarrow in(p(a))\ \}.$$

Viewing Π as a revision program, let

$$D_R = \{s(a), s(b), p(a), p(b)\}.$$

Then, we obtain

$$\Pi_{D_R}|D_I: \quad in(s(a)), out(q(a)) \leftarrow, \\ in(s(b)), out(q(b)) \leftarrow, \\ in(p(a)) \leftarrow,$$

and the necessary change for $\Pi_{D_R}|D_I$ is $(I, O) = (\{p(a), s(a), s(b)\}, \{q(a), q(b)\})$. Hence,

$$D_R = (D_I \cup I) \setminus O$$

holds, and D_R is a Π-justified revision of D_I.

On the other hand, viewing Π as a transaction program, $S = \{in(p(a))*, in(s(a)), out(q(a))\}$ is included in $\mathbf{T}_\Pi \uparrow \omega$. By ignoring marking in S, we obtain $(I', O') = (\{p(a), s(a)\}, \{q(a)\})$. Hence,

$$D' = (D_I \cup I') \setminus O' = \{s(a), s(b), p(a), p(b)\},$$

which coincides with D_R in this case.

Note in the above example that if we do not ignore marking, we get $(IN(S), OUT(S)) = (\{s(a)\}, \{q(a)\})$, which is a smaller explanation of G. Furthermore, our framework can treat extensional facts as abducibles, thereby extends revision programming with *view*.

Hence, our transformation of P into τP provides a method of generating a revision program from the original program P. In deductive databases, such a program P is constructed from IDB. In other application domains, a program P can be any background knowledge base including causal relations, constraints, general laws and commonsense. As far as the authors know, there have been no other methods to generate revision programs automatically, and few discussion has been done on how to get revision programs.

5 CONCLUSION

This paper presented a method of computing extended abduction. An abductive logic program was transformed to a transaction program which declaratively specifies non-determinism in abduction. Then abductive explanations can be computed by the fixpoint of a transaction program, which is sound and complete for acyclic ALPs. In the context of databases, a transaction program provides a declarative specification of database update, and enables us to compute view update in a constructive manner. A transaction program is a kind of disjunctive programs and fixpoint computation is realized by model generation procedures.

This paper assumed acyclic and covered NLPs for programs containing variables. Future work includes relaxing these conditions and exploiting further applications of extended abduction.

References

S. Abiteboul (1988). Updates, a new frontier. In: *Proceedings of the 2nd International Conference on Database Theory, Lecture Notes in Computer Science*, 326, pages 1–18, Springer.

K.R. Apt and M. Bezem (1991). Acyclic programs. *New Generation Computing*, 9:335–363.

C. Aravindan and P. Baumgartner (1997). A rational and efficient algorithm for view deletion in databases. In: *Proceedings of the 1997 International Symposium on Logic Programming*, pages 165–179, MIT Press.

F. Bry (1990). Intensional updates: abduction via deduction. In: *Proceedings of the 7th International Conference on Logic Programming*, pages 561–575, MIT Press.

F. Buccafurri, T. Eiter, G. Gottlob and L. Leone (1997). Enhancing symbolic model checking by AI techniques. IFIG Research Report 9701, Institut für Informatik, Universität Gießen.

K.L. Clark (1978). Negation as failure. In: H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 119–140, Plenum Press, New York.

L. Console, D.T. Dupre and P. Torasso (1991). On the relationship between abduction and deduction. *Journal of Logic and Computation*, 1:661–690.

J. Fernández, J. Grant and J. Minker (1996). Model theoretic approach to view updates in deductive databases. *Journal of Automated Reasoning*, 17:171–197.

M. Gelfond and V. Lifschitz (1988). The stable model semantics for logic programming. In: *Proceedings of the 5th International Conference and Symposium on Logic Programming*, pages 1070–1080, MIT Press.

J. Grant, J. Horty, J. Lobo and J. Minker (1993). View updates in stratified disjunctive databases. *Journal of Automated Reasoning*, 11:249–267.

K. Inoue and C. Sakama (1995). Abductive framework for nonmonotonic theory change. In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 204–210, Morgan Kaufmann.

K. Inoue and C. Sakama (1996). A fixpoint characterization of abductive logic programs. *Journal of Logic Programming*, 27:107–136.

A.C. Kakas and P. Mancarella (1990). Database updates through abduction. In: *Proceedings of the 16th International Conference on Very Large Databases*, pages 650–661, Morgan Kaufmann.

K. Konolige (1992). Abduction versus closure in causal theories. *Artificial Intelligence*, 53:255–272.

N. Lavrač and S. Džeroski (1994). *Inductive Logic Programming: Techniques and Applications.* Ellis Horwood.

V.W. Marek and M. Truszczyński (1994). Revision specifications by means of programs. In: *Proceedings of the 4th European Conference on Logics in AI, Lecture Notes in Artificial Intelligence*, 838, pages 122–136, Springer.

D. Poole (1988). A logical framework for default reasoning. *Artificial Intelligence*, 36:27–47.

R. Reiter (1987). A theory of diagnosis from first principles, *Artificial Intelligence*, 32:57–95.

F. Rossi and S.A. Naqvi (1989). Contributions to the view update problem. In: *Proceedings of the 6th International Conference on Logic Programming*, pages 398–415, MIT Press.

V.S. Subrahmanian (1987). On the semantics of quantitative logic programs. In: *Proceedings of the 4th IEEE Symposium on Logic Programming*, pages 173–182, IEEE Computer Society Press.

A. van Gelder, K.A. Ross and J.S. Schlipf (1991). The well-founded semantics for general logic programs. *Journal of ACM*, 38:620–650.

The KR System dlv: Progress Report, Comparisons and Benchmarks

Thomas Eiter
Institut für Informatik
Universität Gießen
Arndtstrasse 2, D-35392
Gießen, Germany

Nicola Leone
Institut für
Informationssysteme
TU Wien
A-1040 Wien, Austria

Cristinel Mateis
Institut für
Informationssysteme
TU Wien
A-1040 Wien, Austria

Gerald Pfeifer
Institut für
Informationssysteme
TU Wien
A-1040 Wien, Austria

Francesco Scarcello
ISI-CNR, c/o DEIS
Università della Calabria
I-87030 Rende, Italy

Abstract

`dlv` is a knowledge representation system, based on disjunctive logic programming, which offers front-ends to several advanced KR formalisms. The system has been developed since one year at the Technical University of Vienna in an ongoing project funded by the Austrian Science Funds. After a report on the current state of the art in the implementation of `dlv` and of its application front-ends, the paper compares `dlv` with other knowledge representation systems. Both a qualitative and a quantitative comparison are carried out. The first compares the representational power of the systems (intended as the ability to represent problems in a natural and simple fashion). The latter compares the performances of the systems. The `dlv` system turns out to be very powerful from the knowledge representation side and, on the other hand, quite good also in computational power.

Keywords: Implemented KR&R Systems: Reports, Comparisons, Evaluations.

1 INTRODUCTION

The extension of logic programming by disjunction and true negation has been pointed out as a necessary requirement for knowledge representation [16, 2]. This view has been confirmed by results which prove that without disjunction, the expressive capability of logic programming is limited such that relevant problems can not be expressed [12, 7]. Moreover, disjunction has been recognized as an important feature for a declarative KR language, which allows to express knowledge in a simple and natural way. For example, the rule

```
genotype(P,T1) v genotype(P,T2) :-
parent(C,P), heterozygot(C,T1,T2).
```

from a blood group knowledge base[1] may express that the genotype of a parent P of a person C is either $T1$ or $T2$, if C is heterozygot with types $T1$ and $T2$.

After a body of work on the theoretical foundations, more recently implementations of nonmonotonic reasoning systems have been tackled [19, 9, 1, 23]. The dlv system is an effort in this vein, which aims at providing a KR tool based on disjunctive logic programming (DLP) under the stable model semantics [15, 16, 21] and, for true negation, under the answer set semantics [16]. The system supports front-ends to KR applications, and can serve as a declarative knowledge programming language.

The general approach to the system has been described in [13]. In this paper, we describe the state of the art of the implementation and present a quantitative and qualitative comparison to similar KR systems. Our results look promising and show that the current implementation is competitive among the fastest systems. Moreover, by its supreme expressive capability, dlv embodies the most convenient and powerful KR system among those considered here.

2 SYSTEM OVERVIEW

The system architecture is shown in Figure 1. The internal system language is function-free DLP with true negation [16] and integrity constraints. The kernel is an efficient engine for computing all or some stable models of a program. Various frontends, i.e. translators for specific applications into the internal language, have been developed on top of it (cf. Section 3). A Graphical User Interface (GUI) provides convenient access to both the system and its front-ends.

The heart of the system is the *Query Processor* (QP). It controls the system execution and – together with the integrated frontends – it performs pre-processing of the input and post-processing of the generated models, respectively. Upon startup, QP reads the (nonground) program and passes it to the *Rules and Graphs Handler* (RGH), which splits it into subprograms. These subprograms are then, together with facts from a database (Oracle or Access files), submitted to the *Intelligent Grounding Module*, which efficiently generates a subset of the grounded input program, which has the same stable models, but is much smaller in general. (For stratified programs, for example, the Grounding already computes the single stable model.)

QP then again invokes RGH, which generates two partitionings into components of the grounded program. They are used by the *Model Generator* (MG) and the *Model Checker*, and enable a modular program evaluation. This often yields a tremendous speedup.

Then, the heart of the computation is performed by the Model Generator and Model Checker. Roughly, the Model Generator produces some "candidates" stable models, whose stability is verified by the Model Checker.

The generation of the stable models of a program LP relies on a monotonic operator $\mathcal{W}_{LP}$ [17] that extends to the disjunctive case the well-founded operator of [26]. It is defined in terms of a proper notion of *unfounded set*. Intuitively, an unfounded set for a disjunctive program $\mathcal{P}$ w.r.t. an interpretation I is a set of positive literals that definitely cannot be derived from $\mathcal{P}$ assuming the facts in I [17].

Briefly, the algorithm works as follows. $\mathcal{W}^{\omega}_{LP}(\emptyset)$ is first computed, which is contained in every stable model. If $\mathcal{W}^{\omega}_{LP}(\emptyset)$ is a total model, it is returned as the unique stable model. Otherwise, moving from $\mathcal{W}^{\omega}_{LP}(\emptyset)$ towards the stable models, a conjunction of literals (called *possibly-true conjunction* in [17]), the truth of which allows to infer new atoms, is assumed true. The computation proceeds by iteratively applying the inflationary version of

[1]Courtesy of Dietmar Seipel.

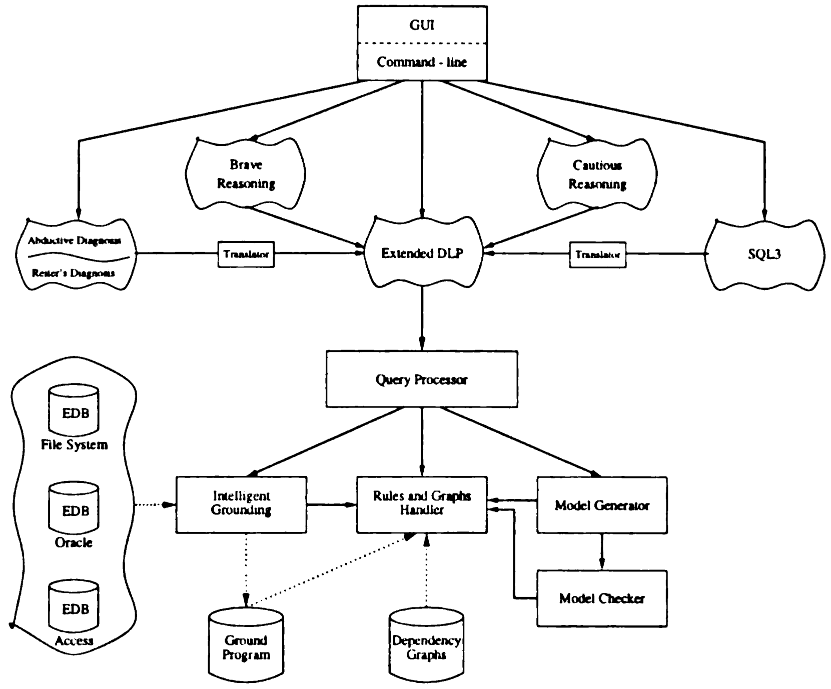


Figure 1: The System Architecture of dlv

the immediate consequence operator, until either a total model of *LP* is reached or two contradictory literals are generated. In the latter case, backtracking is performed; in the former case, the Model Checker is invoked.

The Model Checker (MC) verifies whether the model at hand is a stable model. This task is very hard in general, because checking the stability of a model is well known to be co-NP-complete (cf. [12]). However, recent studies [3, 4, 17] unveiled that minimal model checking — the hardest part of stable model checking— can be efficiently performed for the relevant class of *head-cycle-free* (*HCF*) programs [3].

The MC satisfies the above complexity bounds. Indeed (a) it terminates in polynomial time on every HCF program; and (b) it always runs in polynomial space and single exponential time. Moreover, even on general (non-HCF) programs, the MC limits the inefficient part of the computation to the modules that are not HCF (Note that it may well happen that only a very small part of the program is not HCF).

Finally, once a stable model has been found, control is returned to QP, which performs post-processing and possibly invokes the MG to look for further models.

More details on the basic algorithms implemented in the system can be found in [17, 13, 10]. For space reasons, we can not describe here ongoing refinements.

3 THE APPLICATION FRONT-ENDS

3.1 THE NATIVE dlv INTERFACE

The native interface of the system offers the full power of the internal language of dlv: function-free disjunctive logic programs extended by allowing both integrity constraints and true negation [16]. As will be evident in Section 4, the presence of these constructs makes our language well suited to easily represent a wide range of problems in a natural and highly declarative way.

A `dlv` rule has the following form:

$$\begin{aligned} L_1 \vee \ldots \vee L_k \quad :- \quad & L_{k+1}, \ldots, L_m, \\ & not\ L_{m+1}, \ldots, not\ L_n \end{aligned}$$

where $1 \leq k \leq m \leq n$ and each L_i is a literal, i.e., an atom $a(t_1, \ldots, t_o)$ or its "classical" negation $\neg a(t_1, \ldots, t_o)$, a being the predicate name of that atom, and each t_i $(1 \leq i \leq o)$ being either a constant or a variable. The convention on names is the one of Prolog.

A *constraint* is like a rule with $k = 0$ and $n \geq 1$. A `dlv` program P is a pair $< R, C >$, where R is a set of `dlv` rules and C a set of constraints. A *stable model* of P is any consistent answer set of R [16] satisfying all constraints in C.

3.2 BRAVE AND CAUTIOUS REASONING

The frontends for brave and cautious reasoning are an extension of the native interface described above. In addition to the program, the user also specifies a query, which is essentially the body of a rule followed by a question mark:

$$b_1, \ldots, b_m, \ \texttt{not}\ b_{m+1}, \ldots, \ \texttt{not}\ b_n \ ?$$

where each b_i $(1 \leq i \leq n)$ is a ground literal.

The brave reasoning frontend is invoked by the `-FB` command line option. If the query Q evaluates to true in at least one stable model M, then the system replies: "Q is bravely true, evidenced by M", i.e., returns a stable model that witnesses this fact; otherwise the answer is "Q is bravely false".

Similarly, the cautious frontend is invoked by `-FC`, and the system answers "Q is cautiously true" if and only if Q is true in all stable models. In case of negative answer, we get: "Q is cautiously false, evidenced by M", where M is a stable model witnessing the falsity of Q.

3.3 DIAGNOSIS FRONTENDS

`dlv` provides two kind of diagnosis frontends: the *abductive diagnosis* frontend [20, 11], and the *Reiter's diagnosis* frontend [22].

Our frontend for *abductive diagnosis* currently supports three different modes: general diagnosis, where all diagnoses are computed, subset minimal diagnosis[2], and single failure diagnosis. These modes are invoked by the command line options `-FD`, `-FDmin` and `-FDsingle`, respectively.

The diagnostic theory obeys the syntax described in Section 3.1. Hypothesis (resp. observations) are lists of atoms (resp. literals) separated by a dot (`.`) and are stored in files whose names carry the extension `.hyp` (resp. `.obs`).

Basically, the diagnostic frontend works as follows: After reading the input, disjunctive rules for guessing possible diagnosis candidates are generated from the hypotheses, while the observations become constraints that forbid the generation of inconsistent diagnoses. In the case of subset minimal diagnosis, further rules are added for pruning non-minimal solutions. Finally, the grounding (and subsequently the MG etc.) are invoked, and for each reported stable model, the corresponding diagnosis is returned.

Example 1 (Network Diagnosis) Suppose that machine a is online in the computer network below, but we observe it cannot reach machine e. Which machines are offline? This can be easily modelled as a diagnosis problem, where the theory is

$$\begin{aligned} reaches(X, X) \ &:- \ node(X), not\ offline(X). \\ reaches(X, Z) \ &:- \ reaches(X, Y), \\ & \quad\ connected(Y, Z), \\ & \quad\ not\ offline(Z). \end{aligned}$$

and the observations and hypotheses are respectively

```
not offline(a).  not reaches(a,e).
```

[2]For positive non-disjunctive theories only.

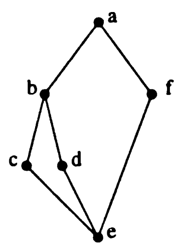


Figure 2: Computer network

and

```
offline(a).offline(b).offline(c).
offline(d).offline(e).offline(f).
```

The frontend for *consistency-based diagnosis* as described by Reiter [22] works similarly. The command line options are `-FR`, `-FRmin` and `-FRsingle`, respectively.

3.4 FURTHER FRONTENDS

The system has a SQL3-frontend based on the current working draft for the standard. It supports recursive query processing such as the bill-of-materials problem.

Other frontends are currently under way. We have started work on a frontend for inheritance reasoning based on the approach in [14]; a frontend for extended logic programs with preferences as in [5] will be tackled in the near future.

4 COMPARISONS AND BENCHMARKS

In this section, we compare `dlv` with two prominent and effective recent systems for nonmonotonic reasoning, DeReS [9] and *Smodels* [19] on a diversified collection of benchmark problems. Basically, we follow the methodology for testing nonmonotonic systems suggested in TheoryBase [8].

4.1 OVERVIEW OF THE COMPARED KR SYSTEMS

Smodels [19] has been recently recognized as the best system to compute stable models for normal (i.e., non-disjunctive) logic programs [25]. Upon request, the system returns the stable models of the input program. *Smodels* supports a kind of brave reasoning by allowing the user to specify a set of atoms that must belong to every returned stable model. It is also possible to specify a maximum number of desired stable models. Thus, *Smodels* also supports nondeterministic semantics: by selecting only one stable model, we look for any nondeterministically chosen stable model of the given program.

DeReS [9] is a KR system implementing classical default logic. It supports several basic reasoning tasks such as testing whether extensions exist, and finding one or all extensions. If the input theory amounts to a logic program, DeReS computes its stable models and supports queries on membership of an atom in some or all stable models. *DeReS* handles only propositional theories. Thus, no uniform problem representation for varying background facts is supported. A preliminary instantiation is needed to emulate programs with variables.

4.2 ENCODING OF BENCHMARK PROBLEMS

Next we present several problems, drawn from different domains, for comparing the KR systems. Some examples have been repeatedly used for system evaluation. However, we also present some novel benchmark problems which are challenging for KR Systems.

Ancestor. Given the parent relationship $parent(\_,\_)$, find the genealogy tree of each person in the database.

This is a classical deductive database example, which has been used to compare the performances of several DDB systems. We consider it to test the suitability of the systems for handling DDB applications and double recursion. For each system, the encoding looks like:

```
ancestor(X,Y) :- parent(X,Y).
ancestor(X,Y) :- ancestor(X,U),
                 ancestor(U,Y).
```

Graph 3-colorability. Given a graph, represented by facts of the form `node(_)` and `edge(_,_)`, assign each node one of three colors such that no two adjacent nodes have the same color.

3-colorability is a classical NP-complete problem, which has been used as a benchmark problem for DeReS [19] and *Smodels* [9]. In *dlv*, the problem can be encoded in a very easy and natural way by means of disjunction and constraints:

```
col(X,r) ∨ col(X,g) ∨ col(X,b) : - node(X).
:- edge(X,Y), col(X,C), col(Y,C).
```

The disjunctive rule nondeterministically chooses a color for each node X in the graph; the constraint enforces that the choices are legal.

In *Smodels*, the problem can be easily encoded as well, but we cannot use either constraints or disjunction, which must be simulated by unstratified negation.

$$\begin{array}{lll} col(X,r) & :- & node(X),\ not\ col(X,g), \\ & & not\ col(X,b). \\ col(X,g) & :- & node(X),\ not\ col(X,r), \\ & & not\ col(X,b). \\ col(X,b) & :- & node(X),\ not\ col(X,g), \\ & & not\ col(X,r). \\ bad & :- & edge(X,Y),\ col(X,C), \\ & & col(Y,C),\ not\ bad. \end{array}$$

Evidently, `dlv` allows for a more elegant problem representation.

The encoding used for DeReS is Δ^3_{col} as described in [8]. For space reasons, we can not list this encoding here.

Prime Implicants. Find the prime implicants of a propositional CNF formula $\Phi = \bigwedge_{i=1}^{n}(d_{i1} \vee \ldots \vee d_{ic_i})$, where the $d_{i,j}$ are literals over the variables $x_1, \ldots, x_m$.

Recall that a *prime implicant* of Φ is a consistent conjunction I of literals such that $I \models \Phi$ and for no subformula I' of I we have $I' \models \Phi$. As easily seen, a prime implicant exists if and only if Φ is satisfiable. Thus, finding a prime implicant of Φ subsumes the satisfiability problem for Φ.

The representation in `dlv` is natural and simple. Here, true negation in the language plays a key role. we have a literal translation of Φ into a `dlv` program P:

$$\mathtt{d}_{11} \vee \ldots \vee \mathtt{d}_{1c_1}. \qquad \ldots \qquad \mathtt{d}_{n1} \vee \ldots \vee \mathtt{d}_{nc_n}.$$

It is easy to see that the stable models of P (cf. Section 3) correspond 1-1 to the prime implicants of Φ.

Smodels has no true negation nor disjunction. We thus rewrite P to a normal logic program as follows.First, each literal $\neg x_i$ is replaced by a new atom x_i', and a constraint

$$\mathtt{bad} \ :- \ \mathtt{x_i},\ \mathtt{x_i'},\ \mathtt{not\ bad}\ .$$

is added, which enforces that no stable model can contain both x_i and x_i', for any $1 \leq i \leq m$. Then any disjunctive rule corresponding to the i-th clause $(1 \leq i \leq n)$

$$\mathtt{d}_{i1} \vee \ldots \vee \mathtt{d}_{ic_i}.$$

is replaced by c_i non-disjunctive rules

$$\begin{array}{lll} d_{il} & :- & not\ d_{i1}, \ldots,\ not\ d_{i(l-1)}, \\ & & not\ d_{i(l+1)}, \ldots,\ not\ d_{ic_i}. \end{array}$$

where $1 \leq l \leq c_i$.

For DeReS, the input theory for Φ has been generated by taking the ground instance for the *Smodels* program, which is then translated into default rules using the translation which maps a rule $\mathtt{A} : -\mathtt{B}_1, \ldots, \mathtt{B}_n, \mathtt{not\ C}_1, \ldots, \mathtt{not\ C}_k$ into the default $\frac{B_1 \wedge \cdots \wedge B_n\ :\ notC_1, \ldots, notC_k}{A}$.

Strategic Companies. The strategic companies problem is from [7]; it is, to the best of our knowledge, the only Σ_2^P-complete KR problem from the business domain. No experimental results for any Σ_2^P-complete KR problems are known.

Briefly, a holding owns companies, each of which produces some goods. Moreover, several companies may have jointly control over another company. Now, some companies should be sold, under the constraint that all goods can be still produced, and that no company is sold which would still be controlled by the holding after the transaction. A company is *strategic*, if it belongs to a *strategic set*, which is a minimal set of companies satisfying these constraints. Those sets are expressed by the following natural program:

```
strategic(C1) v strategic(C2) :-
          produced_by(P,C1,C2).
strategic(C) :-
          controlled_by(C,C1,C2,C3),
          strategic(C1),
          strategic(C2),
          strategic(C3).
```

Here strategic(C) means that C is strategic, produced_by($P, C1, C2$) that product P is produced by companies $C1$ and $C2$, and controlled_by($C, C1, C2, C3$) that C is jointly controlled by $C1, C2$ and $C3$; we have adopted here from [7] that each product is produced by at most two companies and each company is jointly controlled by at most three other companies.

The problem is to find out the set of all strategic companies (i.e., under brave reasoning, for which C the fact strategic(C) is true).

Note that this problem can not be expressed by a fixed normal logic program uniformly on all collections of facts produced_by($p, c1, c2$) and controlled_by($c, c1, c2, c3$) (unless NP = Σ_2^P, an unlikely event). The only system capable of this problem besides ours is DeReS, provided that a default theory is constructed ad hoc by instantiation for an instance.

For DeReS, we have translated the above dlv program into a default theory based on the translation tr_1 [18] (which is appropriate for our purposes). We have also tried a variant of the translation which uses CWA defaults.

Smodels and similar systems can not express **Strategic Companies**. They must use, by current methods, ad hoc programs for particular instances, which need in general exponential time for construction.

4.3 EXPERIMENTAL RESULTS

The results of our experiments are displayed in Figure 6. They were run on a SUN Ultra 1/143 under Solaris 2.5.1, using gcc 2.7.2. The inputs for **Prime Implicants** and **Strategic Companies** were randomly generated; all systems received the same random input.

Since DeReS cannot handle non-ground theories, we generated ground theories using the TheoryBase tool [8]. The time for this grounding is not included in the figures, as it is difficult to assess. It is based on disk saving, which takes on average about one minute total time for larger theories, of which about 15% is CPU time. Thus, the actual total time for DeReS is higher.

For **Ancestor**, we used instances in which the ancestor relation is as depicted in Figure 3. The x-axis of the diagram in Figure 6 shows the number n of people, while the y-axis shows the total time. It appears that both dlv and *Smodels* show a similar time behavior, but *Smodels* has a steeper increase; for 250 people, dlv is doing about 25% faster than *Smodels*. Note that **Ancestor** is mainly a grounding and forward chaining benchmark; as DeReS has no variables, it is not considered here.

For **Graph 3-Colorability**, we have considered two classes of graphs, namely *ladder graphs* and *simplex graphs* (see Figure 4 and Figure 5). The results (see Figure 6) show that *Smodels* and dlv have a similar behavior, but the increase for dlv is slower; for ladder graphs, it runs about 40% faster and for simplex graphs, about 50% faster; moreover, it scales up better to large graphs. DeReS is com-

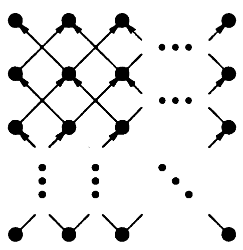


Figure 3: Parent relation

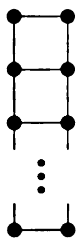


Figure 4: Ladder Graph

parable to **dlv**, but could only handle instances up to about 1100 nodes resp. 650 nodes in total, while the system capacity was exceeded on larger ones. If auxiliary files generated by TheoryBase (whose purpose is not evident) are removed, then DeReS is running considerably slower on both problems.

For **Prime Implicants**, we have randomly generated CNF formulas on n variables using the system of Selman and Kautz [24], where the ratio between the number of clauses and the number of variables is 4.3 (these tend to be hard SAT instances).

Here, *Smodels* has unbeatable performance which nearly parallels the x-axis. dlv has reasonable performance up to $n = 25$, but then the time increases rapidly. (Note that we do not have a smooth curve, since the instances are random.) This may be explained by current weaknesses of dlv in backtracking in the process of model generation. The implementation of intelligent backtracking strategies and modular evaluation is under way (and will be

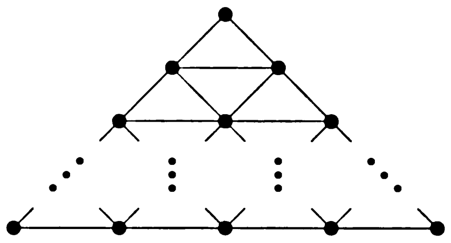


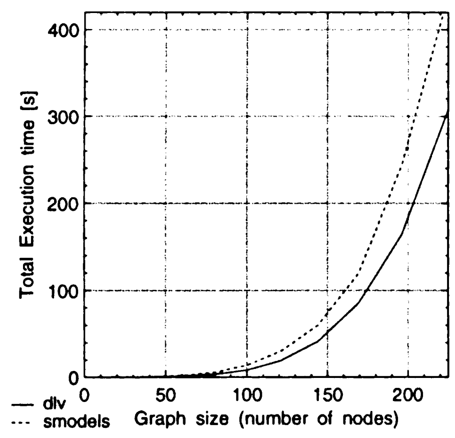
Figure 5: Simplex Graph

completed soon), which will improve the performance significantly. Surprisingly, DeReS has very long running time with steep increase, which is not plotted here. We have no explanation for this.

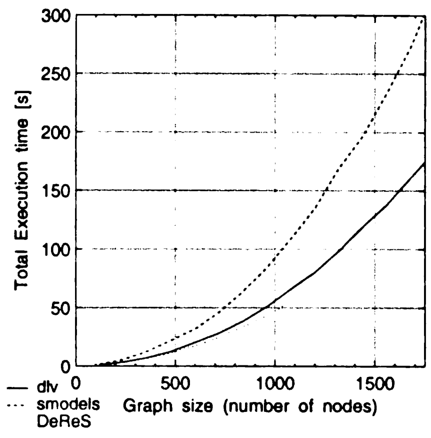
For **Strategic Companies**, we have generated two kinds of tests, each with instances for n companies and $3n$ products.

For the first test, whose results are reported in the left graph, we uniform randomly chose the **produced_by** and the **controlled_by** relations, where each company is controlled by a single joint consortium obeying the obvious constraint that no company can control itself. In the second test, reported in the right graph, each company is controlled by one to five companies (Two consortia have to have at least one member in common), where again the actual number of companies is uniform randomly chosen. On average there are 1.5 **controlled_by** relations per company.

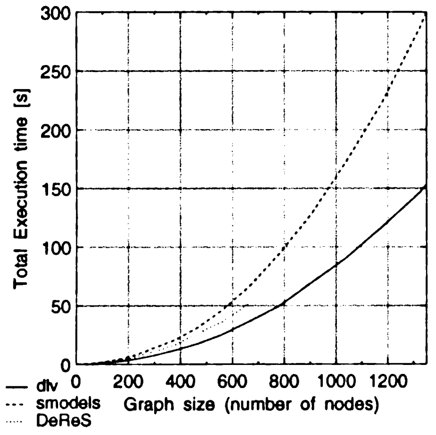
As mentioned above, **Strategic Companies** is a problem which cannot be represented in *Smodels*, and in DeReS only via preliminary instantiation. The results for dlv are interesting. Up to roughly 50 companies, the evaluation in the first case is very fast, and then the running time increases similar as in case of **Prime Implicants**. In the second case performances are somewhat worse and it seems that very hard instances can be met even with fewer companies. We believe that further similar tests may prove useful to better understand



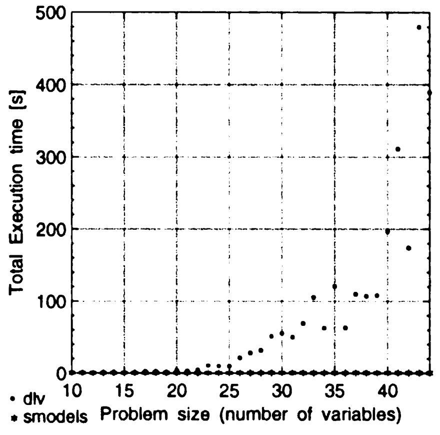
Results for **Ancestor**



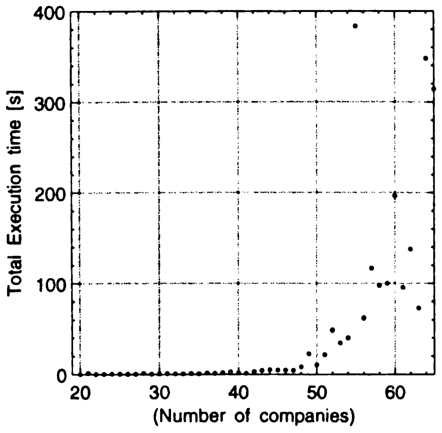
Results for **3-COL Ladder Graphs**



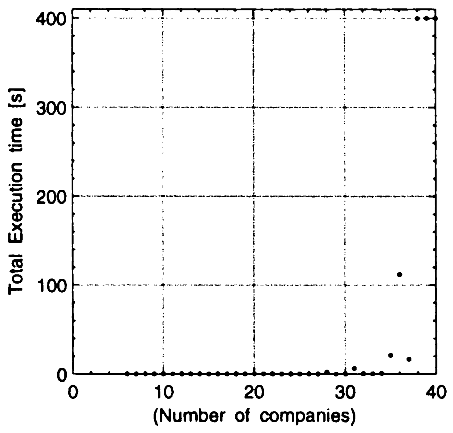
Results for **3-COL Simplex Graphs**



Results for **Prime Implicants**



Results for **Strategic Companies**



Results for **Strategic Companies**

Figure 6: Experimental Results

the complex nature of Σ_2^P-hard problems.

For DeReS, we observed very long running times with steep increase, and only small instances could be handled. The results are not included in the figure.

5 DISCUSSION AND CONCLUSION

The encoding of benchmark problems in the various KR systems, reported in the previous section, highlights quite clearly that `dlv` is the most powerful system from the perspective of knowledge representation power.

All problems we considered are represented in `dlv` in a simpler and more natural fashion than in the other systems. This result confirms the usefulness of extending logic programming by disjunction, true negation and integrity constraints. Roughly, disjunction and integrity constraints allowed an elegant and highly declarative encoding of **3-Colorability**, and true negation played a key role for the representation of **Prime Implicants**. Moreover, the encoding of **Strategic Companies** was possible only thanks to the presence of disjunction (indeed, this problem can not be solved by *Smodels*).

Even if `dlv` is young (the project started only one year ago), and still misses some optimization techniques (that are currently under development), on average it compared positively to the other systems also from the viewpoint of efficiency. Indeed, `dlv` showed the best performances on **Ancestor** and on **3-Coloring** for simplex graphs; it was evidently slower only in the case of **Prime Implicants** (and, even in this case, only compared to *Smodels*).

Importantly, `dlv` was the only system able to handle **Strategic Companies**. This example is important since it definitely requires the use of disjunction in knowledge representation, and can not be expressed without. Thus, the experimental results that we obtained are also interesting from the respect what running time behaviors the need for disjunction as a modelling construct may invoke. As it appeared, up to some realistic instance size, the problem could be handled in reasonable time. This indicates that the expressive power of our `dlv` system is quite convenient: We may express the problem on all possible inputs, without bothering about restrictions on the problem instance under which we *can* write a program (like in *Smodels*), while the actual running time on instances is still acceptable. Further experiments in this direction will be taken.

The experiments show that DeReS can be profitably used when it can exploit the encoding provided by TheoryBase. Indeed, on both "3-coloring" examples (where we used the TheoryBase encoding for DeReS) the system was comparable to `dlv`. However, without the TheoryBase encoding (that was not available on **Prime Implicants** and **Strategic Companies**) the system was very slow. Moreover, even on **3-Coloring**, it was not able handle large instances.

In our opinion, this is because DeReS is a propositional system and has been thought to be used with TheoryBase. *Smodels* confirmed to be a very solid implementation of the stable model semantics for normal (non-disjunctive) programs. On **Ancestor** and both **3-Coloring** its performances were comparable to `dlv` (only slightly worse), and *Smodels* was the most efficient system on **Prime Implicants**. The latter problem evidences that `dlv` is not yet efficient on all domains and that the ongoing work on *modular evaluation* and *intelligent backtracking* is definitely needed to make the system very solid.

Currently, we are improving the system along the lines of [10], in particular, the Model Generator, which will boost the performance on problems like **Prime Implicant** and problems with unstratified negation. Another important ongoing activity concerns the development of efficient data structures that will support linear time implementation of our disjunctive operators. Moreover, we are working also on linguistic extensions, adding weak constraints to the system [6]. This will increase the expressive power of the system beyond Σ_2^P, which enables us to represent problems of higher complexity (in particular, diagnostic examples). Furthermore, these constraints allow for a more natural problem representation in many cases. In addition, we are pursuing the development of further KR frontends, which exploit `dlv` as a declarative knowledge representation system.

For up-to-date information and a download of the system as well as some further examples, please visit the project homepage at `http://www.dbai.tuwien.ac.at/proj/dlv/`.

Acknowledgements

The work was partially supported by *FWF (Austrian Science Funds)* under the project P11580-MAT "A Query System for Disjunctive Deductive Databases" and by the Italian MURST under the project for Italy-Austria cooperation *Advanced Formalisms and Systems for Nonmonotonic Reasoning.*

The authors are deeply indebted to Wolfgang Faber, who designed and implemented the automated benchmark tool for `dlv`, and helped in conducting the experiments.

References

[1] C. Aravindan, J. Dix, and I. Niemelä. Dislop: A research project on disjunctive logic programming. *AI Communications*, 10(3/4):151–165, 1997.

[2] C. Baral and M. Gelfond. Logic Programming and Knowledge Representation. *Journal of Logic Programming*, 19/20:73–148, 1994.

[3] R. Ben-Eliyahu and R. Dechter. Propositional Semantics for Disjunctive Logic Programs. *Annals of Mathematics and Artificial Intelligence*, 12:53–87, 1994.

[4] R. Ben-Eliyahu and L. Palopoli. Reasoning with Minimal Models: Efficient Algorithms and Applications. In *Proceedings Fourth International Conference on Principles of Knowledge Representation and Reasoning (KR-94)*, pages 39–50, 1994.

[5] G. Brewka. Preferred Answer Sets. In *Proceedings Workshop on Logic Programming and Knowledge Representation, at ILPS '97*, Port Jefferson, October 1997.

[6] F. Buccafurri, N. Leone, and P. Rullo. Strong and Weak Constraints in Disjunctive Datalog. In *Proceedings of the 4th International Conference on Logic Programming and Non-Monotonic Reasoning (LPNMR '97)*, Dagstuhl, Germany, July 1997.

[7] M. Cadoli, T. Eiter, and G. Gottlob. Default Logic as a Query Language. *IEEE Transactions on Knowledge and Data Engineering*, 9(3):448–463, May/June 1997.

[8] P. Cholewiński, V. W. Marek, A. Mikitiuk, and M. Truszczyński. Experimenting with Nonmonotonic Reasoning. In *Proceedings of the 12th International Conference on Logic Programming (ICLP-95)*, pages 267–281, Shonan Village Center, Japan, June 1995. MIT Press.

[9] P. Cholewiński, V. W. Marek, and M. Truszczyński. Default Reasoning System DeReS. In *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR '96)*, Cambridge, Massachusetts, USA, 1996.

[10] S. Citrigno, T. Eiter, W. Faber, G. Gottlob, C. Koch, N. Leone, C. Mateis, G. Pfeifer, and F. Scarcello. The `dlv` System: Model Generator and Application Frontends. In *Proceedings of the 12th Workshop on Logic Programming (WLP '97), Research Report PMS-FB10*, pages 128–137, München, Germany, September 1997. LMU München.

[11] T. Eiter, G. Gottlob, and N. Leone. Abduction From Logic Programs: Semantics and Complexity. *Theoretical Computer Science*, 189(1–2):129–177, December 1997.

[12] T. Eiter, G. Gottlob, and H. Mannila. Disjunctive Datalog. *ACM Transactions on Database Systems*, 22(3):315–363, September 1997.

[13] T. Eiter, N. Leone, C. Mateis, G. Pfeifer, and F. Scarcello. A Deductive System for Nonmonotonic Reasoning. In J. Dix, U. Furbach, and A. Nerode, editors, *Proceedings of the 4th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR '97)*, number 1265 in Lecture Notes in AI (LNAI), Berlin, 1997. Springer.

[14] M. Gelfond and A. Gabaldon. From Functional Specifications to Logic Programms. In *Proceedings of the International Logic Programming Symposium (ILPS '97)*, pages 355–369, 1997.

[15] M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In *Logic Programming: Proceedings Fifth Intl Conference and Symposium*, pages 1070–1080, Cambridge, Mass., 1988. MIT Press.

[16] M. Gelfond and V. Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing*, 9:365–385, 1991.

[17] N. Leone, P. Rullo, and F. Scarcello. Disjunctive stable models: Unfounded sets, fixpoint semantics and computation. *Information and Computation*, 135(2):69–112, June 1997.

[18] W. Marek and M. Truszczyński. Stable semantics for logic programs and default theories. In *Proceedings NACLP-89.* MIT Press, 1989.

[19] I. Niemelä and P. Simons. Efficient Implementation of the Well-founded and Stable Model Semantics. In *Proceedings of the 1996 Joint International Conference and Symposium on Logic Programming*, pages 289–303, Bonn, Germany, Sept. 1996.

[20] D. Poole. Explanation and Prediction: An Architecture for Default and Abductive Reasoning. *Computational Intelligence*, 5(1):97–110, 1989.

[21] T. C. Przymusinski. Stable Semantics for Disjunctive Programs. *New Generation Computing*, 9:401–424, 1991.

[22] R. Reiter. A Theory of Diagnosis From First Principles. *Artificial Intelligence*, 32:57–95, 1987.

[23] D. Seipel and H. Thöne. DisLog – A System for Reasoning in Disjunctive Deductive Databases. In *Proceedings International Workshop on the Deductive Approach to Information Systems and Databases (DAISD'94)*, 1994.

[24] B. Selman and H. Kautz, 1997. URL `ftp.research.att.com/dist/ai`.

[25] M. Truszczyński. Automated reasoning with non-monotonic logics (invited talk). In *Proceedings of the 4th International Conference on Logic Programming and Non-Monotonic Reasoning (LPNMR '97)*, pages 112–114, 1997.

[26] A. van Gelder, K. Ross, and J. Schlipf. The Well-Founded Semantics for General Logic Programs. *Journal of the ACM*, 38(3):620–650, 1991.

Disjunctive Ordered Logic: Semantics and Expressiveness

Francesco Buccafurri*
DIMET
Univ. di Reggio Calabria
89100 Reggio Cal., Italy
bucca@ns.ing.unirc.it

Nicola Leone
Information Systems Dept.
Technical Univ. of Vienna
1040 Vienna, Austria
leone@dbai.tuwien.ac.at

Pasquale Rullo
Dip. di Matematica
Univ. della Calabria
87030 Rende, Italy
rullo@si.deis.unical.it

Abstract

The paper presents a knowledge representation language, which extends logic programming with disjunction, inheritance and true negation. The resulting language is called Disjunctive Ordered Logic, $\mathcal{DOL}$ for short. A model-theoretic semantics for $\mathcal{DOL}$ is given, and it is shown to extend the stable model semantics of disjunctive logic programs. A number of examples show the suitability of $\mathcal{DOL}$ for knowledge representation and commonsense reasoning. In particular, the combination of disjunction, inheritance and true negation appears to be a powerful tool for the description of diagnostic processes which are based on stepwise refinements. The complexity and the expressiveness of the language is carefully analyzed. The analysis pays particular attention to the relative power and complexity of inheritance, negation and disjunction; the results support in choosing an appropriate fragment of the language that fits the needs in practice.

*Please address correspondence to Francesco Buccafurri, Fax: +39 965 875220; Phone: +39 965 875302

Keywords. Reasoning Techniques: Efficiency Measures and Complexity; Representational Formalisms: Nonmonotonic Logics; Disjunctive Datalog; Expressive Power.

1 INTRODUCTION

In recent years, there has been a growing interest around the issues of knowledge representation (KR) and non-monotonic reasoning (NMR). NMR tasks, in particular, have received a considerable deal of attention both from the AI and the database community, as the capability of performing non-monotonic reasoning has been recognized to be a primary building block of the future generation intelligent information systems.

A main issue in the knowledge representation (KR) and non-monotonic reasoning (NMR) areas is the problem of designing suitable formalisms for expressing reasoning tasks. Desiderata for these formalisms include good naturalness, high expressive power, modularity, efficiency. As a result, many formalisms have been proposed in the recent past for NMR (e.g., [18, 19, 26, 23, 20], to name a few).

In this paper we present Disjunctive Ordered Logic, $\mathcal{DOL}$ for short, an advanced logic-based formalism for knowledge representation and reasoning. Syntactically, $\mathcal{DOL}$ is an extension of Datalog [28] (function-free definite

Horn clauses) and includes the following constructs: 1) *Disjunction*: disjuncts can appear in $\mathcal{DOL}$ clauses' heads allowing the user to naturally express various forms of incomplete knowledge [2]; 2) *True negation*: the user is given the mean to explicitly state the falsity of atoms (obviously, negation can be used also in rules' bodies); 3) *Modularization*: Knowledge can be structured into components making the implementation of complex reasoning tasks much easier; 4) *Inheritance*: Components can be organized into an inheritance hierarchy encoding relative trustability; this endows the language with a tool for naturally expressing defeasible and multiagent knowledge.

$\mathcal{DOL}$ can be seen either as an extension of Disjunctive Datalog$^{\neg}$ [8, 17, 21] with modularization and true negation (and inheritance), or as an extension of ordered logic ($\mathcal{OL}$) [15, 14, 16, 5] with disjunction. Thus, $\mathcal{DOL}$ features the advantages of both Disjunctive Datalog$^{\neg}$ and $\mathcal{OL}$: (i) the increased expressive power and the possibility to handle incomplete information (w.r.t. $\mathcal{OL}$) because of disjunction and (ii) the enhanced representation capabilities (w.r.t. Disjunctive Datalog$^{\neg}$) due to true negation, modularization and inheritance. In particular, the combination of disjunction, inheritance, and true negation makes the language well-suited also to represent diagnostic processes based on stepwise refinements, that cannot be naturally expressed either in Disjunctive Datalog$^{\neg}$ or in $\mathcal{OL}$.

The main contributions of this paper are the following: 1) We formally define Disjunctive Ordered Logic, providing a model-theoretic semantics based on the concept of *stable model*. (Note that the definition of the semantics for $\mathcal{DOL}$ is much harder than for $\mathcal{OL}$, because of the the interactions of disjunction with both true negation and inheritance in contradiction handling). 2) We prove that Disjunctive Datalog$^{\neg}$ (under the stable model semantics [10, 22, 8]) can be regarded as a fragment of $\mathcal{DOL}$, as every Disjunctive Datalog$^{\neg}$ program can be elegantly mapped into an equivalent $\mathcal{DOL}$ program. 3) We provide a number of examples on the use of $\mathcal{DOL}$ for knowledge representation and defeasible reasoning. For instance, we show how $\mathcal{DOL}$ appears to be a powerful tool for the description of diagnostic processes which are based on stepwise refinements. 4) We carry out a detailed analysis of the expressiveness and complexity of reasoning in $\mathcal{DOL}$. The analysis pays particular attention to the relative power and complexity of inheritance, negation and disjunction. An interesting result in this course is that inheritance does not increase the complexity of the language whereas it does increase its expressive power. For instance, brave reasoning for $\mathcal{DOL}[\neg]$ (i.e., the fragment including true negation but without inheritance and disjunction) is NP-complete, but it does not express *all* NP problems (we show that it cannot express even some simple polynomial problems); if we allow also inheritance, then the complexity is the same but the expressive power increases to capture the whole class NP. The results support in choosing an appropriate fragment of the language that fits the needs in practice.

Related results on complexity and expressive power of KR languages are reported in [8, 12, 6, 19, 25, 24]. Interesting works considering priorities among rules appeared very recently in [3, 11, 30].

2 THE $\mathcal{DOL}$ LANGUAGE

2.1 $\mathcal{DOL}$ BY EXAMPLES

In this section we give an informal presentation of the main capabilities of $\mathcal{DOL}$ for knowledge representation and nonmonotonic reasoning. A $\mathcal{DOL}$ program is a partially ordered set of components, each consisting of a number of rules. These may have disjunctive heads, possibly with negative literals. The partial order can be interpreted as an inheritance hierarchy among components, so that rules flow down from upper to lower components. In case of conflicts, the more specific rule, i.e., the one lower in the hierarchy, provides more reliable information.

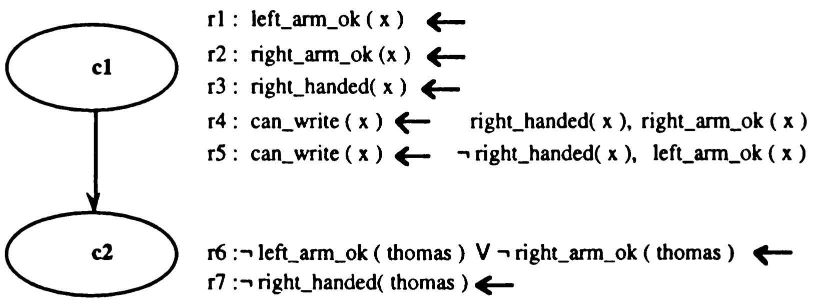

c1
c2
r1 : left_arm_ok (x) ←
r2 : right_arm_ok (x) ←
r3 : right_handed(x) ←
r4 : can_write (x) ← right_handed(x), right_arm_ok (x)
r5 : can_write (x) ← ¬ right_handed(x), left_arm_ok (x)
r6 :¬ left_arm_ok (thomas) V ¬ right_arm_ok (thomas) ←
r7 :¬ right_handed(thomas) ←

Figure 1: the program $\mathcal{P}_{thomas}$

Example 1 (modified from [2])) We want to describe the following situation: (1) normally, a person can use both his arms and is right-handed; moreover, a person can write if he is right-handed (resp., left-handed) and his right (resp., left) arm is unbroken; (2) we just saw *Thomas* with one broken arm but do not remember which; furthemore, we know that *Thomas* is left-handed. The above knowledge is encoded in the $\mathcal{DOL}$ program $\mathcal{P}_{thomas}$ of Figure 1. Here, default knowledge (point 1 above) is represented in the higher component c_1, whose rules express conditions that normally hold. The exceptions to the default knowledge are stated in the lowest component c_2 (formally, we say that $c_2 \leq c_1$ – see Section 2.2). Clearly, knowledge described in the higher component in the hierarchy flows down into the lower one. Now, the question is: Can *Thomas* write or not? Note that, since *Thomas* is left-handed (rule r_7) he can write if his left arm is unbroken (otherwise we don't know, see rule r_5). Because of the uncertainty deriving from our incomplete knowledge about the state of *Thomas*'s arms (rule r_6), we cannot answer the question definitely. Rather, two *possible* answers can be constructed: (a) "Thomas's right arm is broken, and he can write," and (b) "Thomas's left arm is broken, and we don't know whether he can write or not". It will become apparent later in this paper that the stable model semantics naturally represents the above situation. ◇

Example 2 We next use $\mathcal{DOL}$ to model an actual reasoning process of a doctor that has to formulate a diagnosis in presence of a symptom called *acute exordium cephalgia.* The $\mathcal{DOL}$ program that we are going to present is incremental to simulate the stepwise refinement of the diagnostic process. In particular, a new information acquired by the doctor (e.g., by a test) causes the addition of a new component (describing the new information) to the program. At first, the doctor considers all possible diagnostic hypothesis which are compatible with the given symptom. For the sake of simplicity, we confine ourselves to consider only three of such hypotesis, namely, 1) *chephalgia due to a trauma* (denoted by T), 2) *chephalgia due to an infectious disease* (denoted by ID), 3) *chephalgia due to a tumor* (denoted by Tu). Under this simplification, the $\mathcal{DOL}$ program modeling this first reasoning step is the program $\mathcal{P}_0$ of Figure 2 whose stable models are $M_1 = \{chephalgia, T\}$, $M_2 = \{chephalgia, ID\}$ and $M_3 = \{chephalgia, Tu\}$. Each stable model clearly represents a possible diagnosis. Now, suppose that, on the basis of the case-history of the patient, the doctor can exclude that a trauma (hypothesis T) has occurred, i.e., $\neg T$ is true. As a consequence, only hypotesis 2 and 3 above (i.e., infectious disease and tumor, respectively) still hold. The $\mathcal{DOL}$ program representing this new situation is program $\mathcal{P}_1$ depicted in Figure 2 (here, a lower component has been added to $\mathcal{P}_0$). Indeed, the stable models of $\mathcal{P}_1$ are $M_4 = \{chephalgia, ID, \neg T\}$ and $M_5 = \{chephalgia, Tu, \neg T\}$, which capture the intended semantics of the program. As a fur-

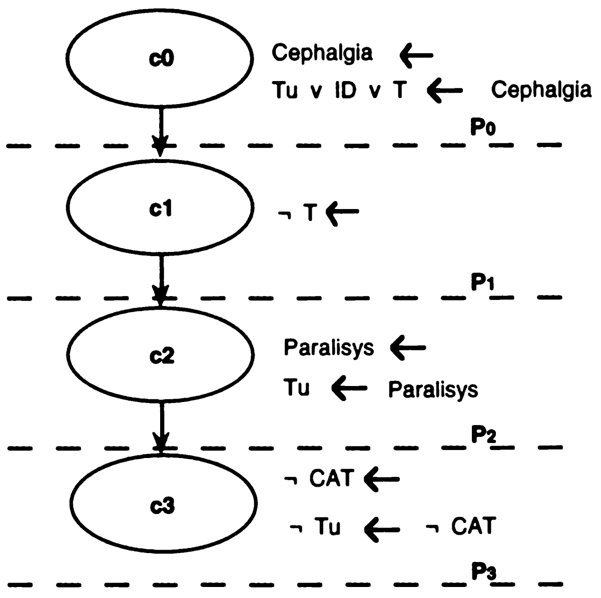


Figure 2: Program $\mathcal{P}$ of example 2

ther step, the doctor performs the *neurological examination*. We suppose that a paralisys of the oculomotor nerve is detected. Then, the doctor assumes the existence of a tumor, as paralysis could be caused by compression. The program modeling this reasoning level is $\mathcal{P}_2$ (see 2) and is obtained from $\mathcal{P}_1$ by adding a lower component consisting of the fact $Paralisys \leftarrow$, that represents the result of the neurological examination, and the rule $Tu \leftarrow Paralisys$, that expresses the conclusion to which the doctor is induced. The stable model semantics for $\mathcal{P}_2$ is given by the unique stable model $M_6 = \{chephalgia, Tu, \neg T, Paralisys\}$, which correctly captures the knowledge of the doctor corresponding to the current reasoning step. Indeed, this knowledge can be summarized as follows: (a) there is no trauma (and $\neg T$ is true in M_6); (b) there is a tumor whose existence has been infered by the presence of a paralisys (and Tu and *paralisys* are both in M_6) and (c) no information on infectious disease ID is available (and neither ID nor $\neg ID$ are in M_6). The lack of a complete information induces the doctor to find a further confirmation to the above conclusion. To this end, he executes the Computerized Axial Tomography (CAT). Surprisingly, the result of this test shows the absence of the tumor. As a consequence, infectious disease becomes the only possible diagnosys. Program $\mathcal{P}_3$ in Figure 2 models this situation as its unique stable model is $M_7 = \{cephalgia, ID, \neg Tu, \neg T, paralisys, \neg CAT\}$ which represents the final diagnosys.◇

Example 3 We want to describe the following situation: You are the organization chair of a big conference. Rules for assigning conference speakers to hotels are the following: (1) Invited speakers are assigned to hotels in the set $\{a, b, c\}$ on a non-deterministic basis; non-invited speakers are assigned to hotels they prefer (as they are paying on their own). (2) You know that some of your italian invited speakers are not capable to speak English, but you do not know exactly who they are; therefore you decide not to assign italian invited speakers with hotel a, where none of the clerks can speak italian.

The above knowledge is represented by the $\mathcal{DOL}$ program of Figure 3. In particular,

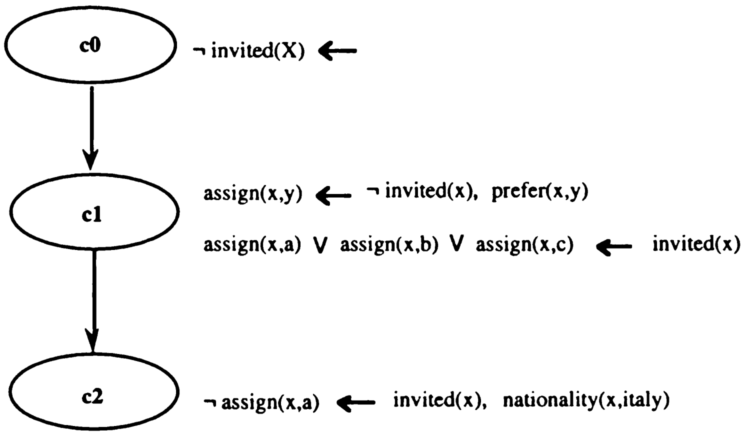

c0
¬ invited(X) ←
c1
assign(x,y) ← ¬ invited(x), prefer(x,y)
assign(x,a) V assign(x,b) V assign(x,c) ← invited(x)
c2
¬ assign(x,a) ← invited(x), nationality(x,italy)

Figure 3: Program $\mathcal{P}_{hotel}$

point 1. above is encoded in the upper component c_1, whereas the constraint in point 2 above is encoded in the component c_2. Note that information regarding invited speakers, hotel preferences, and nationalities, which are not explicitly provided in Figure 3, are assumed to be encoded by a number of facts in the component c_2. Moreover, observe that component c_0 encodes the closed world assumption on the predicate *invited*, whereby the negation occurring in the body of the first rule of c_1 realizes a negation by default.◇

The above examples should have provided hints about the naturalness of $\mathcal{DOL}$ in expressing structured reasoning tasks. In particular, we have shown how, by including mechanisms for the direct description of exception and defeasible reasoning, we have obtained a remarkable improvement in usability, clarity and modeling capabilities of our language w.r.t. Disjunctive Datalog$^{\neg}$, without increasing its computational complexity (see Section 3).

2.2 SYNTAX OF $\mathcal{DOL}$ PROGRAMS

A *term* is either a constant or a variable (note that function symbols are not considered in this paper). An *atom* is $a(t_1, ..., t_n)$, where a is a *predicate* of arity n and $t_1, ..., t_n$ are terms. A *literal* is either a *positive literal* p or a *negative literal* $\neg\, p$, where p is an atom. Two literals are *complementary* if they are of the form p and $\neg p$, for some atom p. Given a literal L, $\neg.L$ denotes its complementary literal. Accordingly, given a set A of literals, $\neg.A$ denotes the set $\{\neg.L \mid L \in A\}$. A *rule* r is a clause of the form $H \leftarrow B$, where H (*head* of the rule) is a disjunction $H_1 \vee ... \vee H_n$ of literals and B (*body* of the rule) is a conjunction $B_1 \wedge ... \wedge B_m$ of literals (note that negative literals may occur in the head of a rule). Given a rule r, we shall denote by $H(r)$ and $B(r)$ the set of literals in the head and in the body of r, respectively. A term, an atom, a literal or a rule is *ground* if no variable appears in it. Let $(C, \leq)$ be a finite partially ordered set of symbols, called *identifiers*. A *component* is a pair $(c, D(c))$, where $c \in C$ and $D(c)$, the *definition of* c, is a finite set of rules. A $\mathcal{DOL}$ *program* (on C) is a set of components, one for each element of C. In the following we shall denote by $<$ the reflexive-reduction of $\leq$ (i.e., $a < b$ iff $a \leq b$ and $a \neq b$). Intuitively, through the $\leq$ relation we model the hierarchical [1] organization of the knowledge, where lower components embody more trustable information.

Example 4 Let $C = \{c_1, c_2\}$, with $c_2 < c_1$, be a set of identifiers. A program on C is $\mathcal{P}_1 = \{(c_1, D(c_1)), (c_2, D(c_2))\}$, where $D(c_1) = \{r_1 : \neg a \leftarrow, r_2 : \neg b \leftarrow\}$, and $D(c_2) = \{r_3 : a \vee b \leftarrow\}$.

2.3 STABLE MODEL SEMANTICS OF $\mathcal{DOL}$

The *Universe* $U_{\mathcal{P}}$ of $\mathcal{P}$ is the set of all constants appearing in the rules of the compo-

[1] Note that, since $\leq$ is a partial order, neither $\leq$ nor $<$ can be ciclic.

nents of $\mathcal{P}$. The *Base* $B_{\mathcal{P}}$ of $\mathcal{P}$ is the set of all possible ground literals constructible from the predicates appearing in the rules of $\mathcal{P}$ and the constants occurring in $U_{\mathcal{P}}$ (clearly, both $U_{\mathcal{P}}$ and $B_{\mathcal{P}}$ are finite). Notice that, unlike disjunctive Datalog$^{\neg}$, the Base of a $\mathcal{DOL}$ program contains also negative literals (as both negative and positive literals are treated in a uniform way).

Given a rule r occurring in a component of $\mathcal{P}$, a *ground instance* of r is a rule obtained from r by replacing every variable X in r by $\sigma(X)$, where σ is a mapping from the variables occurring in r to the constants in $U_{\mathcal{P}}$. We denote by *ground*$(\mathcal{P})$ the (finite) multiset of the ground instances of the rules occurring in the components of $\mathcal{P}$. We can define a function *compOf* from *ground*$(\mathcal{P})$ onto the set C of the identifiers, associating with a ground instance $\overline{r}$ of r the (unique) component of r. Two ground rules r_1 and r_2 are *conflicting on* L if $L \in H(r_1)$ and $\neg.L \in H(r_2)$. An *interpretation* I for a $\mathcal{DOL}$ progam $\mathcal{P}$ is a subset of $B_{\mathcal{P}}$ such that $I \cap \neg.I = \emptyset$. Given an interpretation I, a ground literal L is *true* (resp., *false*) in I if $L \in I$ (resp., $L \in \neg.I$). A literal $L \in B_{\mathcal{P}}$ which is neither true nor false w.r.t. I is said to be *undefined* in I. The head of a rule r is *true* in the interpretation I if $H(r) \cap I \neq \emptyset$; the body of r is *true* in I if $B(r) \subseteq I$. A rule $r \in$ *ground*$(\mathcal{P})$ is *satisfied* in I if its head is true in I or its body is not true in I. Given a $\mathcal{DOL}$ program $\mathcal{P}$, an interpretation $I \subseteq B_{\mathcal{P}}$ is *total* if $I \cup \neg.I = B_{\mathcal{P}}$ (i.e., each literal is either true or false w.r.t. I), otherwise is *partial*. Next we introduce the concept of *model*. The notion of satisfiability of a rule is not sufficient, as it does not take into account the possible presence of explicit contradictions. Hence, we have to introduce the following preliminary definitions.

Definition 1 Let I be an interpretation. Let r_1 and r_2 be two rules in *ground*$(\mathcal{P})$. We say that r_1 *defeates* r_2 *on the literal* L *in* I if: 1) $r1$ and r_2 are conflicting rules on L, and $\neg.L \in H(r_1)$, 2) $compOf(r_2) < compOf(r_1)$ does not hold, and 3) $\neg.L \in I$ and $B(r_1) \subseteq I$. A rule $r \in$ *ground*$(\mathcal{P})$ is *defeated in* I, if for each $L \in H(r)$, there exists $r_1 \in$ *ground*$(\mathcal{P})$ such that r_1 defeates r on L in I. $\diamond$

Intuitively, defeating defines the possible ways to solve conflicts deriving from the presence in the program of contradictory pieces of information.

Example 5 Consider the program $\mathcal{P}_1$, of Example 4. Here, we note that rule r_3 is conflicting with r_1 on a and with r_2 on b, and that r_3 is more specific (in the inheritance hierarchy) than both r_1 and r_2 (as $c_2 < c_1$). Consider now the interpretation $I = \{a, \neg b\}$. We have the following: (i) r_3 defeats r_1 on $\neg a$, and (ii) r_3 does not defeat r_2, as $b \in I$. $\diamond$

Definition 2 Let I be an interpretation for $\mathcal{P}$. We say that I is a *model* for $\mathcal{P}$ if each rule $r \in$ *ground*$(\mathcal{P})$ is satisfied or defeated in I. $\diamond$

Example 6 The interpretation $I = \{a, \neg b\}$ is a model for the program $\mathcal{P}_1$ of Example 4. Indeed, r_2 and r_3 are satisfied and r_1 is defeated in I. $\diamond$

Definition 3 Let I be an interpretation for $\mathcal{P}$. The *reduction of* $\mathcal{P}$ *w.r.t.* I is $P^I = \{r \in$ *ground*$(\mathcal{P}) \mid r$ is not defeated in I and $B(r) \subseteq I\}$. $\diamond$

We note that the reduction of a program is simply a set of ground rules. Given a set S of ground rules, we denote by *horn*(S) the positive disjunctive program (called the *positive version* of S), obtained from S by considering each negative literal $\neg q(\overline{X})$ as a positive one with predicate symbol $\neg q$. Moreover, we denote by MM(*horn*(S)) the set of minimal models [21] of the program *horn*(S).

Definition 4 Let M be a model for a $\mathcal{DOL}$ program $\mathcal{P}$. We say that M is a $\mathcal{DOL}$ *stable model* (or simply "stable model") of $\mathcal{P}$, if $M \in$ MM(*horn*(P^M)) (i.e., M is a minimal model of *horn*(P^M)) $\diamond$

Example 7 Consider the model $M = \{a, \neg b\}$ for the program $\mathcal{P}_1$ of example 4. Now, the reduction of $\mathcal{P}_1$ w.r.t. M is $\mathcal{P}_1^M = \{r_2 : \neg b \leftarrow,\ r_3 : a \vee b \leftarrow\}$ as r_1 is defeated in

M while r_2 and r_3 are not. The positive version of $\mathcal{P}_1^M$ has the following minimal models [21]: $\{b, \neg b\}$ and $\{a, \neg b\}$. Thus, M is a stable model. Similarly, it can be shown that also $\{\neg a, b\}$ is a stable model of $\mathcal{P}_1$. ◇

3 EXPRESSIVE POWER AND COMPLEXITY

3.1 PRELIMINARIES ON EXPRESSIVE POWER

In the context of deductive databases, it is customery to divide the predicate symbols into two sets: EDB and IDB. Some of the predicate symbols correspond to database relations (the *extensional (EDB) predicates*), and are not allowed to occur in rule heads; the other predicate symbols are called *intensional (IDB) predicates*. Actual database relations are formed on a fixed countable domain U, from which also possible constants in a $\mathcal{DOL}$ program are taken.

More formally, a $\mathcal{DOL}$ program $\mathcal{P}$ on a set of identifiers C has associated a relational database scheme $\mathcal{DB}_{\mathcal{P}} = \{r|\ r$ is an EDB predicate symbol of $\mathcal{P}\}$; thus EDB predicate symbols are seen as relation symbols. A database D on $\mathcal{DB}_{\mathcal{P}}$ is a set of finite relations on U, one for each r in $\mathcal{DB}_{\mathcal{P}}$, denoted by $D(r)$; note that D can be seen as a first-order structure whose universe consists of the constants occurring in D (the *active domain* of D).[2] The set of all databases on $\mathcal{DB}_{\mathcal{P}}$ is denoted by $\mathbf{D}_{\mathcal{P}}$.

Given an identifier $c_b \notin C$, we define the program $\mathcal{P}_D$ on $C_b = C \cup \{c_b\}$ by adding to $\mathcal{P}$ a new component $(c_b, D(c_b))$ where: $D(c_b) = \{r(\bar{t}) \leftarrow\ |r \in \mathcal{DB}_{\mathcal{P}} \wedge \bar{t} \in D(r)\}$, and for each $c \in C$, $c < c_b$.

We will analyze the power of $\mathcal{DOL}$ and of its syntactic restrictions, where negation ($\neg$), disjunction ($\vee$), or inheritance ($<$) may be disallowed. Given $X \subseteq \{\neg, \vee, <\}$, we denote by $\mathcal{DOL}[X]$ the fragment of $\mathcal{DOL}$ where the constructs in $(\{\neg, \vee, <\} - X)$ are disallowed. For simplicity, we will omit the braces. For instance, $\mathcal{DOL}[\vee]$ denotes the positive single component (i.e., inheritance-free) $\mathcal{DOL}$ programs, while $\mathcal{DOL}[\neg, \vee, <]$ coincides to full $\mathcal{DOL}$.

Definition 5 A (*bound* $\mathcal{DOL}$) *query* $\mathcal{Q}$ is a pair $\langle \mathcal{P}, G\rangle$, where $\mathcal{P}$ is a $\mathcal{DOL}$ program and G is a ground literal (the *query goal*). Given a database D in $\mathbf{D}_{\mathcal{P}}$, the *answer* of $\mathcal{Q}$ on D is *true* if G belongs to *some* stable model of $\mathcal{P}_D$ (i.e., if G is in the union of the stable models of $\mathcal{P}_D$) and *false* otherwise.[3]

$\mathcal{Q} = \langle \mathcal{P}, G\rangle$ is a $\mathcal{DOL}[X]$ query, where $X \subseteq \{\neg, \vee, <\}$, if $\mathcal{P}$ is a $\mathcal{DOL}[X]$ program.◇

The constants occurring in $\mathcal{P}_D$, G define the active domain of query $\mathcal{Q} = \langle \mathcal{P}, G\rangle$ on the database D.

Observe that, in general, two queries $\langle \mathcal{P}, G\rangle$ and $\langle \mathcal{P}, \neg G\rangle$ on the same database need not give symmetric answers. That is, if e.g. $\langle \mathcal{P}, G\rangle$ answers yes for D, it may be possible that also $\langle \mathcal{P}, \neg G\rangle$ answers yes for D.

A bound query defines a Boolean C-generic query of [1], i.e., a map from $\mathbf{D}_{\mathcal{P}}$ to $\{true, false\}$. As common, we focus in our analysis of the expressive power of a query language on generic queries, which are those mappings whose result is invariant under renaming the constants in D with constants from U.

Definition 6 Let $\mathcal{Q} = \langle \mathcal{P}, G\rangle$ be a (constant-free) query. Then the *database collection* of $\mathcal{Q}$, denoted by $\mathcal{EXP}(\mathcal{Q})$ is the set of all databases D in $\mathbf{D}_{\mathcal{P}}$ for which the answer of $\mathcal{Q}$ is true.

The *expressive power* of $\mathcal{DOL}[X]$ ($X \subseteq \{\neg, \vee, <\}$), denoted $\mathcal{EXP}(\mathcal{DOL}[X])$ is the

[2] We use here active domain semantics (cf. [1]), rather than a setting in which a (finite) universe of D is explicitly provided [9, 7, 29]. Note that Fagin's Theorem and all other results to which we refer remain valid in this (narrower) context; conversely, the results of this paper can be extended to that setting.

[3] We consider brave (also called possibility) semantics in this paper; however, complexity and expressiviness of cautious (also called skeptical) semantics can be easily derived from it.

family of the database collections of all $\mathcal{DOL}[X]$ queries, i.e.,

$\mathcal{EXP}(\mathcal{DOL}[X]) =$
$\{\mathcal{EXP}(\mathcal{Q}) | \mathcal{Q}$ is a constant-free $\mathcal{DOL}[X]$ query$\}\diamond$

The expressive power of each language will be related to database complexity classes, which are as follows. Let C be a Turing machine complexity class (e.g., P or NP), $\mathbf{R}$ be a relational database scheme, and $\mathbf{D}$ be a set of databases on $\mathbf{R}$.[4] Then, $\mathbf{D}$ is *C-recognizable* if the problem of deciding whether $D \in \mathbf{D}$ for a given database D on $\mathbf{R}$ is in C. The *database complexity class DB-C* is the family of all C-recognizable database collections. (For instance, *DB*-P is the family of all database collections that are recognizable in polynomial time). If the expressive power of a given language coincides with some class *DB-C*, we say that the language *captures C*.

3.2 RESULTS

For space limits, we include only some selected proofs, the list of all complexity and expressiveness results is reported in Table 1 and discussed in Section 4.

3.2.1 General $\mathcal{DOL}$ Programs

Lemma 1 *Let $\mathcal{P}$ be a ground $\mathcal{DOL}$ program, and let M be an interpretation. Deciding whether M is a stable model of $\mathcal{P}$ is in* coNP.

Proof. We prove that the complementary problem is in NP. The interpretation M is not a stable model for $\mathcal{P}$ iff either 1) M is not a model for P^M or 2) it is a model for P^M but not minimal, that is, there exists a model I for P^M such that $I \subset M$. Both checking that M is not a model for P^M and checking that a guessed model I for P^M is a proper subset of M are performed in polynomial time. Hence, deciding whether M is not a stable model for $\mathcal{P}$ lies in coNP. $\diamond$

Theorem 1 $\mathcal{EXP}(\mathcal{DOL}[\neg, \vee, <]) \subseteq DB\text{-}\Sigma_2^P$.

[4]As usual, adopting the data independence principle, it is assumed that $\mathbf{D}$ generic, i.e., it is closed under renamings of the constants in U.

Proof. We have to prove that for any $\mathcal{DOL}[\neg, \vee, <]$ query $\mathcal{Q} = \langle \mathcal{P}, G \rangle$ recognizing whether a database D is in $\mathcal{EXP}(\mathcal{Q})$ is in Σ_2^P. D is in $\mathcal{EXP}(\mathcal{Q})$ iff there exists a stable model M of $\mathcal{P}_D$ such that $G \in M$. To check this, we may guess an interpretation M of $\mathcal{P}_D$ and verify that: (i) M is a stable model of $\mathcal{P}_D$, and (ii) $G \in M$. From Lemma 1 (i) is done by a single call to an NP oracle – this task is equivalent to minimal model checking – (note that, since $\mathcal{Q}$ is fixed, *ground*$(\mathcal{P}_D)$ has size polynomial in D, and can be constructed in polynomial time), and (ii) is clearly polynomial. Hence, this problem is in Σ_2^P. Consequently, recognizing whether a database D is in $\mathcal{EXP}(\mathcal{Q})$ is in Σ_2^P.$\diamond$

To show that $\mathcal{DOL}$ expresses *all* queries in Σ_2^P, we demonstrate that disjunctive Datalog$^\neg$ (under stable model semantics) can be simulated in $\mathcal{DOL}$. The result will follow from known results on the expressiveness of disjunctive Datalog$^\neg$ [8].

Given a disjunctive Datalog$^\neg$ program Φ, we represent it as a two-level ordered program *ord*(Φ) called the *ordered version* of Φ. *ord*(Φ) consists of two components, with identifiers *program* and CWA, such that *program* $< CWA$. The definition of *program* is made of the rules in Φ, while the definition of CWA contains a rule of the form $\neg q(X_1, ..., X_n)$ for each predicate q appearing in Φ, where n is the arity of q and $X_1, ..., X_n$ are distinct variables (the above rule is called the *closed world assumption* for q). The intuition behind such a representation is the following: the higher component CWA corresponds to an explicit closed world declaration establishing that a fact is false unless explicitly contradicted.

Note that, by definition of *ord*(Φ), also EDB predicates appear in rule heads' of the CWA component. This can be easily avoided by adding a rule $a'(\bar{X}) \leftarrow a(\bar{X})$ for each EDB predicate occuring in Φ, where a' is a new predicate, and by replacing each occurrence of a in Φ by a'. This way, CWA contains a rule $\neg q(\bar{X}) \leftarrow$, for each IDB predicate q and a rule $\neg a'(\bar{X}) \leftarrow$, for each EDB predicate a. However, since the presence of EDB predicates in CWA does not affect the correctness of following results, for the sake of simplicity, we do not consider the above transformation.

Given an interpretation M for a $\mathcal{DOL}$ program, we denote by M^+ the set of all the positive literals of M and by M^- the set of all negative literals in M.

Theorem 2 *Let Φ be a disjunctive logic program and $ord(\Phi)$ its ordered version. A total interpretation M is a stable model of Φ if and only if M is a $\mathcal{DOL}$ stable model of $ord(\Phi)$*

Proof (*Sketch*) *If-Part.* Let M be a $\mathcal{DOL}$ stable model of $ord(\Phi)$. By definition, it is a minimal model of $horn(ord(\Phi)^M)$. We next show that M is a minimal model of the GL-transform Φ^M of Φ for M and thus it is a stable model for Φ [10]. It is easy to see that M is a model for Φ^M[5] (each rule of Φ^M with a true body is also present in $ord(\Phi)^M$ because it cannot be defeated). To show that M is minimal for Φ^M, we proceed by contradiction, assuming that there exists a model M_1 for Φ^M such that $M_1^+ \subset M^+$. If so, we can derive that the total interpretation M_2, whose positive part coincides with M_1^+, is a model for $horn(ord(\Phi)^M)$ and thus M is not minimal for $horn(ord(\Phi)^M)$ (a contradiction).

(*Only-If Part*). Let M be a stable model of Φ. It is easy to see that M is a model for $horn(ord(\Phi)^M)$. To show minimality (for $horn(ord(\Phi)^M)$), by contradiction we assume that there exists a model M_1 for $horn(ord(\Phi)^M)$ such that $M_1^+ \subset M^+$. If so, we can derive that the total interpretation M_2, whose positive part coincides with M_1^+, is a model for Φ^M and thus M is not minimal for Φ^M (a contradiction). $\diamond$

Example 8 Let $\Phi = \{a \vee b \leftarrow\}$ be a disjunctive logic program. Its stable models are $M_1 = \{a, \neg b\}$ and $M_2 = \{\neg a, b\}$. The ordered version $ord(\Phi)$ of Φ is the $\mathcal{DOL}$ program $\mathcal{P}_1$ of Example 4. We have seen in Example 7 that M_1 and M_2 are the stable models of $\mathcal{P}_1$ too.

[5]Note that we represent total intepretations as sets of literals also for disjunctive Datalog .

Observe that the CWA component is strictly needed; in other words $ord(\cdot)$ cannot be the identity function. Indeed, if take Φ as it is, by looking at it as a single component $\mathcal{DOL}$ program, then it has stable models $M_1' = \{a\}$ and $M_2' = \{b\}$. These models are semantically different from M_1 and M_2. For instance, b is false w.r.t. M_1 while it is not false (in particular, it is undefined) w.r.t. M_1'. $\diamond$

Theorem 3 $\mathcal{EXP}([\neg, \vee, <]) = DB\text{-}\Sigma_2^P$.

Proof. It follows from Theorem 2 and the results in [8].

3.2.2 $\mathcal{DOL}[\neg]$

From the simulation of Datalog$^\neg$ in $\mathcal{DOL}$ defined in Section 3.2.1, it follows easily that $\mathcal{DOL}[\neg, \vee]$ captures NP. In this section we prove that inheritance plays a crucial role in the expressive power of $\mathcal{DOL}$. Indeed, if we disallow inheritance, then $\mathcal{DOL}[\neg]$ is not able to express *all* NP properties (even if it can express *some* of them).

Proposition 1 *Let $\mathcal{P}$ be a $\mathcal{DOL}[\neg]$ program. If the algorithm of Figure 4 outputs M on input $\mathcal{P}$, then M is a stable model of $\mathcal{P}$.*

Proof. Let I be an output of the algorithm of Figure 4 on input $\mathcal{P}$. I is a model of $\mathcal{P}^I$; otherwise, $unsat(\mathcal{P}, I)$ would be non-empty. Let $J \subset I$. Consider the first literal $q \notin J$ which has been included in I in the computation of the algorithm of Figure 4 generating I. The rule $r \in unsat(\mathcal{P}, I)$ which motivated the introduction of q is neither satisfied nor defeated w.r.t. J. Therefore, J is not a model of $\mathcal{P}$ (and of $\mathcal{P}^I$ also).$\diamond$

Corollary 1 *Every $\mathcal{DOL}[\neg]$ program has at least one stable model.*

Proof. The algorithm of Figure 4 produces an output on every $\mathcal{DOL}[\neg]$ program. Therefore, the statement follows from Proposition 1. $\diamond$

Proposition 2 *Let $\mathcal{P}$ be a $\mathcal{DOL}[\neg]$ program. If M is a stable model of $\mathcal{P}$, then there exists*

Input: A $\mathcal{DOL}[\neg]$ program $\mathcal{P}$.
Output: A stable model of $\mathcal{P}$.

Notation: Given a program $\mathcal{P}$ and an interpretation I, $unsat(\mathcal{P}, I)$ denotes the set of rules in $ground(\mathcal{P})$ which are neither defeated nor satisfied w.r.t. I (i.e., $r \in unsat(\mathcal{P}, I)$ iff: (a) $B(r) \subseteq I$, (b) $H(r) \cap I = \emptyset$, and (c) r is not defeated w.r.t. I).

```
begin
        $I := \emptyset$;
        while $unsat(\mathcal{P}, I) \neq \emptyset$ do
                choose a rule $r$ from $unsat(\mathcal{P}, I)$;
                $I := I \cup H(r)$;
        endwhile;
        output $I$;
end.
```

Figure 4: *A Naive Algorithm for Computing Stable Models of* $\mathcal{DOL}[\neg]$

a computation of the algorithm of Figure 4 which outputs M on input $\mathcal{P}$.

Theorem 4 *Let $\mathcal{P}$ be a $\mathcal{DOL}[\neg]$ program and D_1 and D_2 be two databases on $\mathcal{DB}_{\mathcal{P}}$ such that $ground(\mathcal{P}_{D_1}) \subseteq ground(\mathcal{P}_{D_2})$. If M_1 is a stable model for $\mathcal{P}_{D_1}$, then there exists a stable model M_2 for $\mathcal{P}_{D_2}$ such that $M_1 \subseteq M_2$.*

Proof. $\mathcal{P}_{D_1}^{M_1} \subseteq \mathcal{P}_{D_2}^{M_1}$. From Proposition 2, there exists a run of the algorithm of Figure 4 which generates M_1 on input $\mathcal{P}_{D_1}$.[6] Take the same choices on the run of the algorithm on $\mathcal{P}_{D_2}$. Then, whatever are the following choices of the algorithm, the model generated will certainly contain M_1 (as the computation is monotonic).◇

Theorem 5 $\mathcal{EXP}(\mathcal{DOL}([\neg])) \subset DB$-NP.

Proof. Let $\mathcal{DB}_{\mathcal{P}}$ be containing a single unary predicate r. We show next that the *even query*, answering true on the instances of r of even cardinality, cannot be expressed by $\mathcal{DOL}[\neg]$. By contradiction, suppose a query $\mathcal{Q} = \langle \mathcal{P}, G \rangle$ expresses the even query and let D_1 be a databases such that $|D_1(r)|$ is even (we denote by $|X|$ the cardinality of set X). Then, there exists a stable model M_1 for $\mathcal{P}_{D_1}$ such that $G \in M_1$. Consider now a database D_2 such that $D_1(r) \subseteq D_2(r)$ and $|D_2(r)| = |D_1(r)| + 1$. Since the hypotheses of Theorem 4 apply, the answer of $\mathcal{Q}$ on D_2 is true too (contradiction).◇

[6]Even if $\mathcal{P}_{D_1}$ has two components, it is equivalent to the program where facts and rules are included in a single component (as no contradiction may arise on database facts).

4 DISCUSSION AND CONCLUSION

The results on the expressiveness and complexity of $\mathcal{DOL}$ are compactly represented in Table 1. Each column of the table collects the results for a specific fragment of $\mathcal{DOL}$; for instance, $\mathcal{DOL}[\neg, <]$ refers to programs where inheritance ($<$) and true negation ($\neg$) are allowed, but disjunction ($\vee$) is not. For complexity, each entry of a complexity class C symbolizes C-completeness; *ens.* means that the existence of a stable model is ensured, and deciding the existence of stable model is therefore trivial. For expressive power, $= C$ means that DB-C is captured (i.e., precisely the database collections in C are expressible); while, $\subset C$ means that only a strict subset of the database collections in DB-C are expressible.

Full $\mathcal{DOL}$ (i.e., $\mathcal{DOL}[\vee, \neg, <]$) has a very high expressive power, as it expresses exactly the

Table 1: Expressiveness and complexity results on $\mathcal{DOL}$

	$\{\}$	$\{<\}$	$\{\neg\}$	$\{\neg,<\}$	$\{\vee\}$	$\{\vee,<\}$	$\{\vee,\neg\}$	$\{\vee,\neg,<\}$
(*a*) *Recognition and Existence of stable models for propositional (ground) programs*								
Recognition	P	P	P	P	coNP	coNP	coNP	coNP
Existence	ens.	ens.	ens.	NP	ens.	ens.	ens.	Σ_2^P
(*b*) *Expressive Power and Data Complexity of Brave Reasoning for general programs*								
Expr. Power	$\subset$ P	$\subset$ P	$\subset$ NP	$=$ NP	$\subset \Sigma_2^P$	$\subset \Sigma_2^P$	$\subset \Sigma_2^P$	$= \Sigma_2^P$
Data Complex.	P	P	NP	NP	Σ_2^P	Σ_2^P	Σ_2^P	Σ_2^P

properties in DB-Σ_2^P. Interestingly, reasoning in $\mathcal{DOL}$ is not harder than reasoning in disjunctive Datalog$^\neg$ (they are both Σ_2^P-complete). Thus, we argue that the inclusion of inheritance and true negation is highly desirable: it yields a remarkable gain as to the clarity, modeling features, and overall usability of the language, and it does not imply any overhead in computational complexity. (It does cause a tangible increase of the expressive power; but, we argue that capturing Σ_2^P is sufficient for most applications.)

The analysis of the fragments of $\mathcal{DOL}$ highlights the importance of coupling inheritance and true negation. True negation alone ($\mathcal{DOL}[\neg]$) augments the complexity of $\mathcal{DOL}[]$ up to NP; but it allows to express only a strict subset of NP. If we add also inheritance, then the complexity remains the same; but we increase the expressive power up to capturing the whole class NP. We observe a similar behaviour, one complexity level upper, for $\mathcal{DOL}[\vee,\neg]$ and $\mathcal{DOL}[\vee,\neg,<]$.

In sum, the results of Section 3 prove the high expressiveness of the $\mathcal{DOL}$ language. Moreover, the examples provided in Section 2.1 show some cases where $\mathcal{DOL}$ can be profitably used for knowledge representation and reasoning. Several important issues merit further investigation. First, we plan to contrast our approach to other related languages that have been presented in the literature [4, 10, 13, 27], Theorem 2 is a first step in this direction. Moreover, it is important to characterize the properties of our semantics. For instance, we would like to see whether the semantic principles stated in [4] are satisfied by our language. Finally, algorithms for the evaluation of $\mathcal{DOL}$ programs should be also designed.

References

[1] S. Abiteboul, R. Hull, and V. Vianu (1989). *Foundations of Databases.* Addison-Wesley.

[2] C. Baral, and M. Gelfond (1994). Logic Programming and Knowledge Representation. *Journal of Logic Programming,* **19-20**:73–148.

[3] G. Brewka, Preferred Answer Sets, *Proc. LPKR'97,* Port Jefferson, New York, October 1997.

[4] Brewka, G., Eiter, T., Preferred Answer Sets for Extended Logic Programs, *Proc. KR'98,* Morgan Kaufmann, 1998.

[5] Buccafurri, F., Leone, N., Rullo, P., Stable Models and their Computation for Logic Programming with Inheritance and True Negation, *Journal of Logic Programming,* Vol 27(1) April 96, pp 5-43.

[6] M. Cadoli and M. Schaerf, A Survey of Complexity Results for Non-monotonic

Logics, *Journal of Logic Programming*, Vol. 17, pp. 127-160, 1993.

[7] Chandra, A., Harel, D., Structure and Complexity of Relational Queries, *Journal of Computer and System Sciences*, 25, 1, 1982, pp. 99-128.

[8] T. Eiter and G. Gottlob and H. Mannila. Disjunctive Datalog, *ACM Transactions on Database Systems*, September, 1997.

[9] R. Fagin. Generalized First-Order Spectra and Polynomial-Time Recognizable Sets. In R. M. Karp, editor, *Complexity of Computation*, pp. 43–74. AMS, 1974.

[10] M. Gelfond and V. Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing*, 9:365-385, 1991.

[11] Gelfond, M. and Son, T.C., Reasoning with Prioritized Defaults, *Proc. of the Workshop Logic Programming and Knowledge Representation - LPKR'97*, Port Jefferson, New York, October 1997.

[12] Gottlob, G., Complexity Results for Nonmonotonic Logics, *Journal of Logic and Computation*, Vol. 2, N. 3, pp. 397-425, 1992.

[13] Kowalski, R.A., Sadri, F., Logic Programs with Exceptions, *Proc. of 7th ICLP*, Jerusalem, 1990, pp. 598-616.

[14] Laenens, E., Saccá, D., and Vermeir, D., Extending Logic Programming, *Proc. of ACM SIGMOD*, May 1990.

[15] Laenens, E., Vermeir, D., A Fixpoint Semantics for Ordered Logic, *Journal of Logic and Computation*, vol.1, n.2, December, 1990, pp. 159-185.

[16] Leone, N., Mecchia, A., Rossi, G., Rullo, P., A Deductive Environment for Dealing with Objects and Non-Monotonic Reasoning, *IEEE-TKDE*, 9(4), 1997.

[17] Leone, N., Rullo,P., Scarcello, F., Disjunctive Stable Models: Unfounded Sets, Fixpoint Semantics and Computation, *Information and Computation*, Academic Press, 135(2), 1997, pp. 69-112.

[18] Marek, W., Truszczyński, M., Modal logic for default reasoning, *Annals of Mathematics and Artificial Intelligence*, 1, 1990, pp. 275-302.

[19] Marek, W., Truszczyński, M., Autoepistemic Logic, *Journal of the ACM*, 38, 3, 1991, pp. 518-619.

[20] McCarthy, J., Circumscription - a Form of Nonmonotonic Reasoning, *Artificial Intelligence*, 13, pp. 27-39, 1980.

[21] J. Minker. On Indefinite Data Bases and the Closed World Assumption. *Proc. CADE-82*, 1982, pp. 292–308.

[22] T. Przymusinski. Stable Semantics for Disjunctive Programs. *New Generation Computing*, 9:401–424, 1991.

[23] Reiter, R., A Logic for Default Reasoning, *Artificial Intelligence*, vol.13, pp. 81-132, 1980.

[24] D. Saccà. The Expressive Powers of Stable Models for Bound and Unbound DATALOG Queries. *Journal of Computer and System Sciences*, Vol. 54, No. 3, June 1997, pp. 441–464.

[25] Schlipf, J.S., The Expressive Powers of Logic Programming Semantics, *Proc. ACM Symposium on Principles of Database Systems* 1990, pp. 196-204.

[26] G. Shvarts, Autoepistemic Modal Logics, Proc. TARK-90, Morgan Kaufman, 1990.

[27] Touretzky, D.S., *The Mathematics of Inheritance Systems*, Pitman London, 1986.

[28] Ullman, J.D., *Principles of Database and Knowledge-Base Systems*, Vol. 1 and 2, Computer Science Press, Rockville, Md., 1989.

[29] Vardi, M.Y., The Complexity of Relational Query Languages, *Proc. ACM Symp. on Theory of Computing*, 1982, pp. 137-146.

[30] Zhang, Y., Foo, N.Y., Answer Sets for Prioritized Logic Programs, *Proc. of the International Symposium* pp. 69-84, Port Jefferson, New York, October 1997.

Planning and Execution

Modeling an Agent's Incomplete Knowledge during Planning and Execution

Fahiem Bacchus
Department of Computer Science
University of Waterloo
Waterloo, Canada N2L 3G1
fbacchus@logos.uwaterloo.ca

Ron Petrick
Department of Computer Science
University of Waterloo
Waterloo, Canada N2L 3G1
rpapetrick@logos.uwaterloo.ca

Abstract

In many domains agents must be able to generate plans even when faced with incomplete knowledge of their environment. We provide a model to capture the evolution of the agent's knowledge as it engages in the activities of planning (where the agent must attempt to infer the effects of hypothesized actions) and execution (where the agent must update its knowledge to reflect the actual effects of actions). The effects (on the agent's knowledge) of a planned sequence of actions are very different from the effects of an executed sequence of actions, and one of the aims of this work is to clarify this distinction. The work is also aimed at providing a model that is not only rigorous but can also be of use in developing planning systems.

1 Introduction

In this paper we address the problem of how agents who must operate in incompletely known environments can generate and execute plans. In particular, we examine the case where an agent has correct but incomplete knowledge of its environment. A representation scheme for incomplete knowledge is developed that is specifically oriented towards the development of actual planning systems. In particular, we focus on representing and updating the kinds of incomplete knowledge that would be useful to a planning agent capable of sensing and manipulating its environment, and we ensure that the representation can be used in a straightforward manner in an actual planning system.

When planning, the agent must reason about the effects of actions. When the agent has complete knowledge of its environment, there is no need to distinguish between what the agent knows and what is true in its environment. Hence, in classical planning systems there is no explicit separation between the agent's knowledge and facts about the world. For example, when a STRIPS database is employed to model the world state it is only implicit that the agent knows the contents of the database.

When faced with incomplete knowledge, however, we do require an explicit model of the agent's knowledge and the manner in which this knowledge is affected by the actions executed by the agent. In fact, for the purposes of planning it is the action's effects on the agent's knowledge that are most important: at plan time the agent must *know* that the plan will achieve its desired effects, and at execution time the agent must have sufficient *knowledge* at every step of the plan to execute it [Lev96].

A major complication, when having to reason about how actions affect the agent's knowledge, arises from the fact that the plan time effects of such actions are quite different from their execution time effects. For example, say that the agent is operating in the UNIX domain and that it is considering the action of listing a directory. At plan time all that it will know is that after the action it will know all of the files in the directory: the actual identity of those files will not become known until the action is actually executed.

In many domains generating plans that operate correctly no matter how the world is configured is impossible—such conditional plans [PS92, PG93] end up being too large. Instead, the agent must often commit by actually executing some actions so as to avoid having to plan for contingencies that never occur. However, execution also has its pitfalls, as executing an action might change the world in such a way that the agent's ultimate goal becomes impossible to achieve.

Understanding how to manage these tradeoffs so that we can effectively interleave planning and execution remains an important open problem in the area. We believe that our work makes a contribution to this problem. In particular, our representation of actions provides a clear separation between their plan time and execution time effects. We can project the agent's knowledge state through both planned actions sequences and executed action sequences. This provides useful information about the differences between plan time and execution time and leads to a deeper understanding of both plan time and execution time effects. It also opens up a wider range of possibilities for interleaving planning and execution.

The general approach we adopt is much like the traditional STRIPS representation. In particular, we use a collection of databases to represent the agent's knowledge. However, we provide a formal semantics for the items in each database. We do this by translating each of these items into formulas of a modal logic of knowledge. Actions operate much like STRIPS actions do: they modify the contents of the various databases. Through examples we show that a useful range of actions can be represented as update operations to these databases.

Our approach allows us to project the agent's knowledge through a sequence of planned actions: we simply apply the actions' plan time effects to the agent's initial knowledge state to produce a sequence of intermediate knowledge states. This means that a straightforward forward chaining search could in principle be used to generate plans. We can also project the agent's knowledge state through sequences of action executions, and this means that an plan execution module can also be supported by our formalism.[1].

In the rest of the paper we will present the method we use for representing the agent's knowledge, discuss how inferences can be made from this knowledge, and briefly discuss how actions are represented and how they update the agent's knowledge. Finally, we will close with some simple examples that show how our approach models the plan time and execution time effects of various actions and plans. But first we discuss some related work.

[1]Some work would have to be done to modify our approach to support partial order planning or backwards chaining planning. This should be possible as such planning technologies were initially developed from "projective" action semantics like ours. However, such approaches are not a major interest of ours, as we are pessimistic about their ultimate future. We are much more optimistic about the future of forward chaining planners [BK96, McD96]

1.1 Related work

The general issue of planning with correct but incomplete knowledge has received a great deal of attention recently. There are many domains that can be usefully modeled under this paradigm. For example, Etzioni, Golden and Weld have been engaged in ongoing research into software agents that operate in the UNIX and Internet environments [EGW97, GW96, EGW94, GEW94]. As they point out, these domains are reasonably approximated by the assumption of correct but incomplete knowledge. The main feature of their work has been to develop methods for providing such agents with planning capabilities: exactly the issue we address here. Their work, particularly their work on locally closed worlds [EGW97] has been very influential in our work.

There are two main differences between their work and that presented here. First, much of their approach is tied to the technology of partial order planning. We feel that this often has the detrimental effect of making the semantics of their representations and algorithms much more difficult to understand. The projective semantics we use here gives a clearer separation between the issues that involve the semantics of actions and the agent's knowledge and the issues that involve the implementation and semantics of partial order planning. The second difference is that their work is intimately tied to execution time effects. For example, the algorithms they develop for reasoning about locally closed world conditions [EGW97] assume that the actions achieving such conditions have been executed. This means that the planning system they construct is forced to interleave planning and execution in an inflexible manner. There is limited scope for alternative ways of interleaving planning and execution, to deal, e.g., with domains where executing actions can produce irreversible changes.

As pointed out by Levesque, there are a number of subtle issues involved in planning in the face of incomplete knowledge. In [Lev96] Levesque provides a formal specification of plan correctness in the face of incomplete knowledge. He points out that plans have knowledge preconditions, and that it must be known at plan time that these conditions will be achieved at execution time. Although Levesque's work provides vital insights into the problem, his work does not directly address the issue of generating plans. In particular, his model of actions and knowledge is specified in the situation calculus. Hence, to reason about the effects of actions one would, in general, have to employ full first-order inference. In our work we have used some of Levesque's ideas about plan correctness, but have focused on more limited representations that can be implemented more effectively in real planning systems.

2 Representing the Agent's Knowledge

The first issue we address is that of representing the agent's knowledge. As mentioned in the introduction we are assuming that the agent has correct, albeit incomplete, information about its environment. This kind of information is conveniently formalized using a standard modal logic of knowledge (see [FHMV95] for an introduction).

One of our aims, however, is to develop an approach that can facilitate the development of effective planning systems, and we do not know, at this time, how to deal with a fully general logic of knowledge. Instead we adopt a STRIPS like approach where by the agent's knowledge is represented as a collection of databases each of which maintains a particular type of knowledge. We formally characterize the agent's knowledge by providing a translation from the database contents to a set of logical formulas. Thus we utilize the logic's semantics as the underlying semantics of our representation.[2] We use **DB** to represent the agent's databases, and **KB** to represent the set of logical formulas that characterize the agent's knowledge.

In brief, the standard modal logic of knowledge adds a modal operator K to an ordinary first-order language, extending the language's syntax by adding the rule: if ϕ is a formula then so is $K(\phi)$. Semantically, the language is interpreted over a collection of worlds W, each of which is an ordinary first-order model. These worlds are related to each other by an accessibility relation. In this case every world is accessible from every other world. Any non-modal formula ϕ is interpreted to be true at a particular world w (written $w \models \phi$) iff it is true according to the standard rules for interpreting first-order formulas. A formula of the form $K(\phi)$ is interpreted to be true at w iff ϕ is true at every world accessible from w, which means that ϕ must be true at every world in W (since at every world all worlds are accessible).

Intuitively, the agent's knowledge is being modeled by the set W. The agent does not know which of the worlds in W is the real world, and considers all of these worlds to be possible versions of the way the real world is configured. If it does not know whether or not ϕ is true, then there will be worlds in W where ϕ is true and worlds where ϕ is false. Knowing ϕ to be true means that ϕ is true in every world in W. Our assumption that the agent's knowledge is correct is modeled by the fact that the real world is a member of W. Thus, if the agent knows ϕ, ϕ is in fact true in the real world. For convenience, we use the notation w^* to represent the real world. Furthermore, when we write a logical formula we always interpret it at w^*. Thus, a formula like $K(readable(kr.tex)) \wedge writable(kr.tex)$ means that the agent knows that file *kr.tex* is readable (and by the semantics of K, *kr.tex* is in fact readable) and that it is in fact writable (but this is not necessarily known by the agent). A useful notation is $K_{\mathbf{whe}}(\phi)$ which is defined to be the formula $K(\phi) \vee K(\neg\phi)$: either ϕ or its negation is known to hold.

2.1 Rigid Terms and Constant Domains of Discourse

The agent's knowledge will include atomic facts about various terms. For example, knowing that the file *kr.tex* is readable might be represented by the atomic formula $K(readable(kr.tex))$, where *kr.tex* is a term of the language. We also allow functions. For example, the agent might know various function values like $K(size(kr.tex) = 1024)$, i.e., *kr.tex* is 1024 bytes in length.

Terms composed from functions and constants, like *kr.tex*, 1024, and *size*(*kr.tex*), pose potential problems when dealing with knowledge. In particular, the terms they generate may be rigid or non-rigid. Non-Rigid terms are terms whose denotation varies from world to world, while rigid terms have a fixed denotation across worlds. For example, the agent might not know the size of the file *kr.tex*, so the term *size*(*kr.tex*) may have a different denotation (i.e., a different value) in the different worlds the agent considers possible. On the other hand a number like 1024 would have the same denotation (i.e., the same meaning) in every world.

When terms can be of either type reasoning about facts like *readable*(*kr.tex*) becomes more complex.[3] For example, it is not immediately obvious what it would mean for the agent to know this fact if the term *kr.tex* had potentially a different denotation in every world. Since there does not seem to be a good reason to have this level of generality, we impose the restriction that all constants must be rigid. Thus, a term like *kr.tex* will always denote the same object in every world.[4] On

[2] In essence we are simply restricting ourselves to a particular subset of the logic.

[3] See Garson [Gar77] for a good discussion of these issues.

[4] There may be many files in the agent's environment called *kr.tex*. In practice, we would have to use a distinct constant for each file. For example, we could use a unique

the other hand, we allow functions to generate non-rigid terms. Thus, a term like *size*(*kr.tex*) can denote a different value in different worlds.

Formally, this means that for every constant c in the language describing any particular planning domain, the agent's knowledge (the set **KB**) includes the formula:

$$\exists x. K(x = c). \tag{1}$$

This says that there is a particular object in the real world such that in every possible world the constant c denotes that object.

We assume that numeric functions, like "+", or numeric predicates like "<" have their standard interpretation in every world (hence they also are rigid).

Another complication that we wish to avoid are those that arise when different worlds w can have different domains of discourse.[5] So we restrict our semantics to only consider models in which all worlds have an identical domain of discourse.[6]

2.2 The Databases

We represent the agent's knowledge by a collection of four databases, each of which is discussed below.

K_f: The first database is much like a standard STRIPS database, except that both positive and negative facts are allowed and we do not apply the closed world assumption. In particular, K_f can include any ground literal (atomic formula or negation of an atomic formula). K_f is further restricted so that all the terms that appear in any literal must be constants. So, for example, an atomic formula like *readable*(..(*dir*)), where the function ".." specifies the parent directory of a direction file, cannot appear in K_f. To include such information we would have to know the name of *dir*'s parent directory.

In addition to literals K_f can also contain specifications of function values. In particular, formulas of the form $f(c_1, \ldots, c_n) = c_{n+1}$, where f is an n-ary function and the c_i are all constants. This formula specifies that f's value on this particular set of arguments is the constant c_{n+1}. In effect, our restriction means that function values in K_f are considered to be known by the agent only if they can be "grounded" out as constant values.

We specify what the contents of K_f means in terms of the agent's knowledge by specifying that for every formula $\ell \in K_f$, **KB** includes the formula:

$$K(\ell). \tag{2}$$

K_w: The second database contains a collection of formulas every instance of which the agent either knows or knows the negation. In particular, K_w can contain any formula that is a conjunction of atomic formulas. By adding simple ground atomic facts to K_w we can model the effects of sensing actions at plan time. In particular, at plan time if the agent hypothesizes executing a sensing action that senses some fact like *readable*(*kr.tex*), all the agent will know is that after sensing it will know whether or not this fact is true. Only at execution time will there be a resolution of this disjunction.

In a similar manner by adding formulas containing variables to K_w we can model the plan time effects of actions that generate universal effects like local closed world information [EGW97]. For example, the UNIX "*ls dir*" command yields local closed world information about the contents of directory *dir*. Yet at plan time the agent will not know the actual contents of the directory. The contents will only become known after the *ls* action is executed.

We specify what the contents of K_w means in terms of the agent's knowledge by specifying that for every formula $\phi(\vec{x}) \in K_w$ (a conjunction of atomic formulas in which the variables in $\vec{x}$ appear free), **KB** includes the formula

$$\forall \vec{x}. K(\phi(\vec{x})) \vee K(\neg\phi(\vec{x})). \tag{3}$$

Note that in the case where $\vec{x}$ is the empty set (i.e., ϕ is a conjunction of ground atomic formulas), this reduces to the formula $K_{\mathbf{whe}}(\phi)$.

Some predicates, e.g. numeric predicates like < and equality =, have the same denotation in every world in W. Such "rigid" predicates are considered to be implicitly in K_w. For example, $x > y$ and $x = y$ are implicit members of K_w. The inference algorithm presented below has access to these implicit members of K_w.

K_v: The third database is simply a specialized version of K_w designed to store information about various function values the agent will come to know. In particular, K_v can contain any unnested function term.

identifier for each file and have a function *name* that maps this identifier to the file's "common" name. The function *name* may thus map many different files to the same common name. However, for readability we will continue to use common names in our examples, leaving it to the reader to remember that all such names are intended to be unique.

[5] Again see [Gar77] for a discussion.

[6] We have not found that this poses any practical problems. In particular, this assumption does not mean that we know the identity of all the objects in the real world.

For example, $f(x, a)$ would be a legal entry in K_v but $f(g(a), c)$ would not be. Like K_w, the entries in K_v can be used to model sensing actions, except in this case the sensors are returning constants (e.g., numbers) not truth values. The value returned by the sensor will not be known until execution time, but at plan time the agent will know that such a value will become known.

For every formula $f(\vec{x}) \in K_v$, where $\vec{x}$ is the set of variables appearing in the term, **KB** includes the formula

$$\forall \vec{x}. \exists v. K(f(\vec{x}) = v). \tag{4}$$

Formulas of this type are a standard way of specifying that the agent knows a function value, see, e.g., [SL93].

More general information about knowing function values can be specified by entries in K_w. For example, if we will come to know the sizes of all the files in a particular directory *dir*, we could place $\mathit{in\text{-}dir}(x, \mathit{dir}) \wedge \mathit{size}(x) = y$ in K_w, where $\mathit{in\text{-}dir}(x, y)$ means that x is in directory y. This formula says that for every file x that is in directory *dir* we know all values of y such that $\mathit{size}(x) = y$. Of course since *size* is a function there is only one such y.

***LCW*:** The fourth database is a database of local closed world information. The innovative concept of locally closed worlds comes from the work of Etzioni et al. [EGW97]. *LCW* represents the execution time analog of K_w, and basically asserts that the agent's K_f database contains a complete list of all items satisfying a particular conjunction of atomic formulas. In most cases such a list can only be added to the K_f database by actually executing an action.

LCW can contain formulas of exactly the same form as K_w: conjunctions of atomic formulas. We specify the semantics of the *LCW* database as follows. Let $\phi(\vec{x}) = \alpha_1(\vec{x}) \wedge \ldots \wedge \alpha_k(\vec{x})$ be a conjunction of atomic formulas in which the vector of variables $\vec{x} = \langle x_1, \ldots, x_n \rangle$ appear free. Say that $\phi \in LCW$.[7] Let $C = \{\vec{c} : \alpha_i(\vec{x}/\vec{c}) \in K_f, 1 \leq i \leq k\}$. C is the set of tuples of constants explicitly listed in K_f as satisfying ϕ. For every such formula $\phi \in LCW$, **KB** includes the formula

$$\forall \vec{x}. \bigwedge_{\vec{c} \in C} \neg(x_1 = c_1 \wedge \ldots \wedge x_n = c_n) \Rightarrow K(\neg\phi(\vec{x}/\vec{c})). \tag{5}$$

For example, if $P(x) \wedge Q(x, y) \in LCW$, and $P(a)$, $P(c)$, $Q(a, b)$ and $Q(a, c)$ are all in K_f, (which means that the pairs (a, b), and (a, c) are explicitly listed as satisfying $P(x) \wedge Q(x, y)$ in K_f), then the formula

$$\forall x, y. \neg(x = a \wedge y = b) \wedge \neg(x = a \wedge y = c) \\ \Rightarrow K\Big(\neg(P(x) \wedge Q(x, y)\Big),$$

is in **KB**. This formula says that the pairs (a, b) and (a, c) are in fact the only pairs satisfying $P(x) \wedge Q(x, y)$. Thus it entails, e.g., that $K(\neg(P(b) \wedge Q(b, c)))$.

This formula makes explicit the notion utilized by Etzioni et al. that if we have local closed world information and we don't have an instance explicitly listed in the database then we can conclude that the property does not hold.

2.3 The semantics of LCW and K_w

We have provided a semantics for the *LCW* and K_w databases by translating their contents to modal logic formulas. In doing this we are using the well understood semantics of the modal logic to provide a final grounding for the entries in these databases. It is useful to point out that when we convert entries in K_w to formulas of the form $\forall \vec{x}. K(\phi(\vec{x})) \vee K(\neg\phi(\vec{x}))$ this corresponds to the agent knowing that the set of satisfying instances of $\phi(\vec{x})$ is invariant across worlds. That is, a tuple of constants $\vec{c}$ satisfies $\phi(\vec{x}/\vec{c})$ in the real world if and only if it satisfies the formula in every world the agent considers possible.

The presence of such a formula in K_w does not mean, however, that the agent knows the truth value of $\phi(\vec{x}/\vec{c})$, since the action that will resolve this has not yet been executed. When the formula is in *LCW* the action has already been executed and all of the satisfying instances of ϕ have been added to the agent's K_f database by the action. Hence, the agent will know the truth value of $\phi(\vec{x}/\vec{c})$ for every $\vec{c}$. Thus a typical action specification will include a plan time addition to K_w and an execution time addition to *LCW*.

The concept of locally closed worlds as a generalization of the closed world assumption is due to Etzioni et al. who develop the concept in detail in [EGW97]. In our approach, however, we have carefully separated local closed world information into plan time effects and execution time effects. The inference algorithm developed in [EGW97] is an execution time algorithm that requires the actions executed to actually add all of the satisfying instances to the K_f database. At plan time the satisfying instances are not yet known, yet we still want to perform "local closed world" reasoning at plan time. Our approach gives us that ability.

[7]Note that not every variable in $\vec{x}$ need appear free in every literal.

2.4 The Knowledge State

Given a particular set of these four databases, i.e., a particular **DB**, the agent's knowledge state is defined by the set of formulas in **KB** as specified by the formulas 1–5 above. In particular, the agent's knowledge state is characterized by the set of models (in which every possible world has the same domain of discourse) that satisfy all of the formulas in **KB**.

It can be shown that subject to obvious consistency requirements any **DB** specifies a consistent **KB**.

Theorem 2.1 *Let* DB *be any set of these four databases subject to the two conditions*

1. *there is no atomic formula α with both α and $\neg\alpha$ in K_f and*

2. *no function $f(c_1, \ldots, c_n)$ is specified to have two distinct values in K_f.*

Then the **KB** *corresponding to* **DB** *is consistent. That is,* **KB** *has a model.*

Proof: In general **KB** will have many models. We show how an arbitrary model can be constructed. First, we let the domain of discourse be the set of all constants appearing in **DB**. Then we construct a single first-order model w by starting with the set of ground literals (and function values) contained in K_f. Then we add to K_f a set of negative facts sufficient to satisfy all of the formulas arising from LCW. Let $\forall\vec{x}. \bigwedge_{\vec{c}\in C} \neg(x_1 = c_1 \wedge \ldots \wedge x_n = c_n) \Rightarrow K(\neg\phi(\vec{x}/\vec{c}))$ be a formula in **KB** arising from a formula $\phi \in LCW$. For every $\vec{c} \notin C$ we pick a conjunct of $\phi(\vec{x}/\vec{c})$, $\alpha_i(\vec{x}/\vec{c})$, that is not in K_f: one such conjunct must exist by the definition of C. In fact, more than one such conjunct may exist, in which case we make an arbitrary choice. We add $\neg\alpha_i(\vec{x}/\vec{c})$ to K_f, thus satisfying that negative instance of ϕ. We do this for every negative instance of every $\phi \in LCW$.

Note that since no positive facts are added to K_f, our additions do not affect what we can infer from LCW. (The sets C of satisfying instances do not change.) Hence, the addition of negative facts to K_f in order to satisfy a formula $\phi \in LCW$ will not affect the additions required to satisfy any other formula $\phi' \in LCW$.

Clearly, the resulting set of facts in K_f continues to satisfy the above two conditions, and thus this set of facts has at least one first-order model. We pick an arbitrary model, w. Finally, we build a model for the modal logic by setting the collection of models W to be simply the set $\{w\}$. It is not difficult to see that this set of worlds W satisfies any formula of the form $\forall\vec{x}.K(\phi(\vec{x})) \vee K(\neg\phi(\vec{x}))$ that could arise from entries in K_w and K_v. ■

Corollary 2.2 *If actions are specified as additions and deletions to these databases and these updates maintain the obvious consistency conditions, then no sequence of actions can give rise to an inconsistent* **KB**.

Intuitively, this theorem says that our representational formalism remains much like the classical STRIPS representation. In STRIPS any database is logically consistent and any sequence of actions maintains this consistency. This is true for our representation as well (except we must outlaw obvious inconsistencies). Like STRIPS this has both positive and negative features. On the positive side, a user of our representation need not worry about "breaking" the representation by generating an inconsistent state. On the negative side, the onus is on the user to build an accurate domain model. As with STRIPS the user must ensure that the **KB** represented by the databases makes sense in the domain being modeled, and that the actions update **KB** in an sensible manner. For example, as with STRIPS, if there are state constraints (e.g., the agent can't be carrying an object and have its hands empty at the same time), then the user must ensure that the databases representing the initial world satisfies those constraints and that the actions properly update the databases so as to maintain those constraints.

3 Inference from DB

From its collection of databases the agent can infer various things. An inference procedure is sound if whenever it infers a formula ϕ from **DB** we have that **KB** $\models \phi$; the procedure is complete if **KB** $\models \phi$ implies that ϕ can be inferred by the procedure from **DB**. Unfortunately, complete inference is impractical, as the set of things that follow from **KB** includes all logical truths (this is the famous problem of logical omniscience [Hin75]).

Fortunately planning applications typically do not require particularly complex reasoning. The major requirement is usually to decide whether or not an atomic formula is true or false at a particular point in a plan. When dealing with incomplete knowledge the requirements become more complex, e.g., we may need to determine whether or not the agent will $K_{\mathbf{whe}}$ some fact at a particular point in a plan. In Table 1 we present a simple procedure for answering queries about atomic formulas from the databases.

Procedure **IA**(ε)
Inputs: Either a ground atomic formula containing the terms $(t_1, \ldots, t_k)$, or a single term. The terms in ε can contain functions but no variables.
Output: **T**, **F**, **W**, or **U** subject to the conditions: (1) **T** implies **KB** $\models K(\varepsilon)$, (2) **F** implies **KB** $\models K(\neg\varepsilon)$, (3) **W** implies **KB** $\models K_{\text{whe}}(\varepsilon)$ (know whether) when ε is a formula and **KB** $\models \exists x.K(x = \varepsilon)$ when ε is a term, and (4) **U** implies the algorithm is unable to conclude anything about ε.

1. Simplify all terms by replacing each t_i in ε by **EvalT**(t_i).
2. If ε is the term t and either (1) t is a constant or (2) there exists a $t' \in K_v$ and a substitution θ such that $t'\theta = t$, then **return(W)**. Else **return(U)**.
3. If ε is of the form $t_1 = t_2$, then if these two terms are syntactically identical **return(T)**. Else if t_1 and t_2 are both constants then **return(F)**. Else **return(U)**.
4. If $\varepsilon \in K_f$, then **return(T)**.
5. If $\neg\varepsilon \in K_f$, then **return(F)**.
6. If there exists a $\phi(\vec{x}) = \alpha_1(\vec{x}) \wedge \ldots \wedge \alpha_k(\vec{x}) \in LCW$ and a ground instance of ϕ, $\phi(\vec{x}/\vec{a})$, such that (1) $\vec{a}$ are constants appearing in K_f, (2) $\alpha_i(\vec{x}/\vec{a}) = \varepsilon$ for some i, and (3) **IA**$(\alpha_j(\vec{x}/\vec{a}))$ = **T** for all $j \neq i$, then **return(F)**.
7. If there exists a $\phi(\vec{x}) = \alpha_1(\vec{x}) \wedge \ldots \wedge \alpha_k(\vec{x}) \in K_w$ and a ground instance of ϕ, $\phi(\vec{x}/\vec{a})$, such that (1) $\vec{a}$ are either constants appearing in K_f or terms t_i appearing in ε for which **IA**(t_i) = **W** , (2) $\alpha_i(\vec{x}/\vec{a}) = \varepsilon$ for some i, and (3) **IA**$(\alpha_j(\vec{x}/\vec{a}))$ = **T** for all $j \neq i$, then **return(W)**.
8. Else return **U**.

Procedure **EvalT**(t)
Inputs: A variable free term.
Output: t' the simplest term known to be equal to t.

1. If t is a constant then **return**(t).
2. If $t = f(t_1, \ldots, t_k)$ and $f(\textbf{EvalT}(t_i), \ldots, \textbf{EvalT}(t_k)) = c \in K_f$ or we can compute that f on these arguments is equal to c (e.g., when f is an arithmetic function) then **return**(c), else **return**(f(**EvalT**(t_i), ..., **EvalT**(t_k))).

Table 1: Inference Algorithm

This algorithm can be shown to be sound. Its complexity is dominated by the search for ground instances of $\phi(\vec{x})$ in steps 4 and 5. Potentially the number of ground instances of $\phi(\vec{x})$ can be exponential in the number of variables in $\vec{x}$. However, we do not feel that this will be an issue in practice.

As an example of the operation of **IA** consider the query **IA**(*size*(*kr.tex*) > 1000) when *size*(*kr.tex*) $\in K_v$ is the only entry in any of the databases. In this case **IA** will return **W**. Intuitively, since the agent will come to know the value of *size*(*kr.tex*) it will also come to know whether or not that size is larger than 1000. First **IA** tries to reduce the function term *size*(*kr.tex*), but no reduction is known as this term is not in K_f. There are no entries in LCW so the algorithm progresses to step 7. The predicate $>$ is rigid and thus $\phi = x > y$ is an implicit entry in K_w (see discussion of K_w above). Since *size*(*kr.tex*) $\in K_v$, **IA**(*size*(*kr.tex*)) = **W** and the ground substitution $\{x = size(kr.tex), y = 1000\}$ satisfies condition (1). Under this substitution condition (2) is satisfied and (3) is trivially satisfied as ϕ has no other conjunctions.

4 Representing Actions

The previous sections have provided a mechanism for representing an agent's knowledge state in a STRIPS like manner as a collection of databases. We have also provided a mechanism for answering some simple queries from these databases. In this section we show how we can model actions in a very STRIPS like manner as well. In particular, the preconditions of actions involve testing the contents of the various databases, and the action effects bottom out on a set of adds and deletes to the databases. This means that starting at some initial configuration of the agent's knowledge state we can decide what actions can be applied and we can compute what the agent's new knowledge state will be after the action has been applied.

A major theme throughout the paper has been the separation between plan time and run time. This separation is maintained in our action descriptions. Every action has a specified set of plan time effects and a set of run time effects. Both plan time and run time effects are encoded as database updates. This means that we can compute the plan time effects of a sequence of actions or track their execution time effects in the same formalism. This will be illustrated by the examples presented in Section 5, but first we specify more formally the representation of actions.

Actions are specified by four components: the parameters, the preconditions, the plan time effects, and the run time effects.

The action's parameters. This is simply a set of variables that can be bound to produce a particular instance of the action.

The action's precondition. Since it is the agent that is executing or planning the actions a decision on whether or not an action can be executed must be based on the agent's knowledge state: the agent has no direct access to the state of its environment. To this end it is possible to develop a query language for querying the status of its databases. However, to keep things simple we will specify preconditions to be a conjunctive set of primitive queries. All queries in the set must evaluate to true to satisfy the precondition. The primitive queries all utilize the above inference algorithm and they are listed below. In this listing α is any ground atomic formula, and t is any variable free term.

1. $K(\alpha)$, true iff $\mathbf{IA}(\alpha)$ returns $\mathbf{T}$.
2. $K(\neg\alpha)$, true iff $\mathbf{IA}(\alpha)$ returns $\mathbf{F}$.
3. $K_w(\alpha)$, true iff $\mathbf{IA}(\alpha)$ returns $\mathbf{W}$, $\mathbf{T}$, or $\mathbf{F}$.
4. $K_v(t)$, true iff $\mathbf{IA}(t)$ returns $\mathbf{W}$.
5. The negation of any of the above four queries.

The action's plan time effects. These are specified by a list of condition effect statements of the form $C \Rightarrow E$. Each condition C is a conjunctive set of primitive queries, and each effect E is a set of additions or deletions to the four databases.

The action's run time effects. We assume a simple interface between the planner and the execution module. In particular, when an action instance is executed the name of that action is passed to the execution module along with a list of "run-time" variables [GW96]. The execution module binds the run-time variables with information it obtains while executing the action.[8] The execution module may generate a sequence of bindings for the run-time variables. The effects of the action are specified using a list of condition effect statements, $C \Rightarrow E$, as before. For run-time effects, however, C and E may contain any of the run-time variables. Furthermore, C may contain tests on the run-time variables. If $C \Rightarrow E$ contains a run-time variable then this condition effect statement will be evaluated once for every distinct binding of the run-time variables generated by the execution module. On the other hand, when $C \Rightarrow E$ has no runtime variables it is only executed once.

Additions and deletions to the four databases are specified by formulas like $add(K_f, size(kr.tex) = 33000)$, which adds this function value to the K_f database. We assume that *add* and *delete* have been configured so as to maintain the obvious consistency conditions mentioned in Theorem 2.1. For example, when we add the function value to K_f we delete any previous function values.

5 Examples

Our first example is that of opening a safe, due originally (we believe) to Moore [Moo85]. There are two actions available: *readComb* and *dialComb*. Formal descriptions of these actions are given in Table 2. We consider two different plans to see if they achieve the goal of opening the safe.

Consider the situation where the agent's initial knowledge state I is described by $K_f = \{haveComb(safe)\}$, i.e., the object "*safe*" has a combination lock. The agent might try dialing a random combination on the safe, for instance, taking the action *dialComb*(*safe*, *15-42-7*). In I it is easy to see that $\mathbf{IA}(haveComb(safe)) = \mathbf{T}$. Furthermore, $\mathbf{IA}(\textit{15-42-7}) = \mathbf{W}$ since "*15-42-7*" is a constant (step 2 of the algorithm) and all constants are known. Hence

[8] The run-time variables are positional just as in a procedure call. The user has to know what information is returned by the execution module at each position in order to properly specify the action.

Command	Precondition	Effects
readComb(x)	$K(haveComb(x))$	**Plan Time:** $add(K_v, combo(x))$ **Run Time:** $exec(readComb(x), !val)$ $delete(K_v, combo(x)), add(K_f, combo(x) = !val)$
dialComb(x, y)	$K(haveComb(x))$, $K_v(y)$	**Plan Time:** $K(y = combo(x)) \Rightarrow add(K_f, (open(x)))$ **Run Time:** $exec(dialComb(x), !safeopen)$ $!safeopen = \textbf{True} \Rightarrow$ $add(K_f, (open(x))), add(K_f, (y = combo(x)))$

Table 2: Open Safe Domain Actions

the agent knows at plan time that the action's preconditions are satisfied.

Since the action's preconditions are satisfied, the action can be simulated[9] on I to yield an updated **DB**, I'. In this case however $I' = I$ since the action has no plan time effects on I. *dialComb* has a conditional plan time effect, but in this case **IA** cannot deduce the condition $K(y = combo(safe))$ from I and so the effect $add(K_f, open(safe))$ is not activated. Intuitively, the agent does not know if dialing a random combination will cause the safe to open.

When we execute the action from the initial state I, however, we get a different set of effects. The combination *15-42-7* is passed to the execution module along with the run time variable *!safeopen* (this is the $exec(dialComb(x, y), !safeopen)$ component of the action where x is bound to *safe* and y is bound to *15-42-7*). The execution module will set *!safeopen* to **True** or **False** dependent on whether or not the action succeeded in opening the safe. At run time, if *!safeopen* is set to **True** by the execution module, the action's conditional effect will be activated resulting in both *open*(*safe*) and *combo*(*safe*) = *15-42-7* being added to K_f to create a new state I'. Intuitively, if the safe opens the agent comes to know it and also comes to know that the combination dialed was in fact the right combination. So we see that the act of dialing a arbitrary combination does not allow the agent to conclude at plan time that the safe will be opened. However, at run time the agent may in fact be lucky and cause the safe to open.

Now consider the action sequence *readComb*(*safe*) followed by *dialComb*(*safe*, *combo*(*safe*)), again from initial state I. The precondition to the first action, *readComb*(*safe*), is satisfied in I. At plan time this action updates I by adding *combo*(*safe*) to K_v. Intuitively, this action will cause the agent to come to know the combination of the safe. Let the updated state be I'.

In I', $K(haveComb(safe))$ holds as this fact was not deleted from K_f. Furthermore, $K_v(combo(safe))$ also holds as this term was added to K_v by the previous action. Thus, we can conclude that the preconditions of the second action *dialComb*(*safe*, *combo*(*safe*)) hold in I'. When we simulate the action in I' we must determine if the conditional of *dialComb*'s plan time effect holds in I'. For this action instance the conditional is $K(combo(safe) = combo(safe))$. I' has nothing in it to allow the inference algorithm to simplify these terms, but the algorithm is still able to return **T** as the two terms are syntactically identical (step 3 of the **IA** algorithm). Hence, *open*(*safe*) is added to the K_f database of I'. Intuitively, the agent knows at plan time that these two actions will open the safe, even though it does not currently know what combination will be dialed.

At run time, *readComb*(*safe*) has the effect of determining what the actual value of the combination is. The execution module binds this value to the run time variable *!val*. Suppose that this value is *15-42-7*. Then *combo*(*safe*) = *15-42-7* will be added to K_f. In addition, the term *combo*(*safe*) is deleted from K_v.[10] These changes will be made to the initial state I to yield a new state I'. Now *dialComb*(*safe*, *combo*(*safe*)) is executed in I'. Prior to passing information to the execution module we must reduce all terms to their simplest form using the **EvalT** algorithm. This means that the run time call to the execution module will be $exec(dialComb(safe, \textit{15-42-7}), !safeopen)$: the second argument of the action *combo*(*safe*) will have been reduced to *15-42-7* by the function value added

[9]We use the term "simulated" when talking about projecting the action's effects at plan time, and "executed" when talking about projecting the action's effects at run time.

[10]This deletion is not strictly necessary. It "cleans up" K_v by removing redundant information.

Command	Effects
drink	**Plan Time:** $add(K_f, hydrated)$
medicate	**Plan Time:** $K(hydrated) \Rightarrow add(K_f, \neg infected)$ $K(\neg hydrated) \Rightarrow add(K_f, dead)$ $\neg K_w(hydrated) \Rightarrow delete(K_f, \neg dead)$ **Run Time:** $exec(medicate, !alive)$ $!alive = \textbf{False} \Rightarrow add(K_f, dead)$ $!alive = \textbf{True} \Rightarrow add(K_f, \neg infected)$
stain	**Plan Time:** $add(K_w, blue), add(K_w, infected)$ **Run Time:** $exec(stain, !stainblue)$ $delete(K_w, blue), delete(K_w, infected)$ $!stainblue = \textbf{True} \Rightarrow add(K_f, blue), add(K_f, infected)$ $!stainblue = \textbf{False} \Rightarrow add(K_f, \neg blue), add(K_f, \neg infected)$

Table 3: Medical Domain Actions

by the previous action. This reduction is important, and is the reason we need a $K_v(y)$ precondition on the *dialComb* action: the execution module cannot be expected to take complex terms whose value is unknown as arguments. If the execution module is successful it will return **True** in the run time variable *!safeopen*, which will cause *open*(*safe*) to be added to K_f in I'. The other addition is redundant as the value of *combo*(*safe*) is already in I'.

Our second example is due to Smith and Weld. Three actions are available: *drink*, *medicate*, and *stain*. The goal is to cure a patients' infection, without killing them. *drink* has the effect of hydrating the patient. *medicate* has the ability to cure the infection, but only if the patient is hydrated. Otherwise, it kills the patient. *stain* can be used to test if the patient is infected: the stain becomes blue if the patient is infected. These actions are described in Table 3. None of these actions have preconditions that need to be satisfied, so we are only concerned with their effects.

Suppose that the agent's initial knowledge state is described by $K_f = \{\neg dead\}$. One possible plan is the action sequence *drink* followed by *medicate*. *drink* has the plan time effect that the agent knows that the patient is hydrated. The second action, *medicate*, has a conditional plan time effect. Since the agent knows *hydrated*, it will also come to know $\neg infected$. Furthermore, $K(hydrated)$ implies $K_w(hydrated)$ so the third conditional is not activated. Hence, neither of these actions removes $\neg dead$ from K_f, so the agent also knows the patient will be alive after these two actions. Thus, the agent is able to construct to plan that it knows will achieve its goals. Furthermore, it knows this at plan time.

Another possible plan is to perform the action *medicate* without first hydrating. Since initially the agent does not have any knowledge about hydration the third conditional effect is activated and the agent loses its knowledge that the patient is not dead. So at plan time the agent can conclude that the medicate action has an unknown effect on *dead* and hence that this plan is not safe.

Finally consider the plan *stain* followed by the conditional action if $K(infected)$ then *drink* followed by *medicate*. The action *stain* has the plan time effect of adding *infected* to K_w. In other words, the agent knows at plan time that after executing *stain* it will either be in a state where it knows *infected* or it knows $\neg infected$. It is not difficult to extend the planner so that at plan time it can add a conditional branch for any fact in K_w, like *infected*. Along one of the branches it adds *infected* to K_f, assuming *infected* to be true, and along the other it adds $\neg infected$ to K_f assuming *infected* to be false. It then proceeds to complete the plan along both branches ensuring that all branches achieve the goal. At execution time the K_w fact that conditions any branch will be resolved and the plan executor will know which branch to take.

In this example, after the *stain* action one branch will start in a state where $K_f = \{\neg dead, infected\}$. In this state it is not difficult to see that the actions *drink* then *medicate* achieve the agent's goal. The other branch starts in a state where $K_f = \{\neg dead, \neg infected\}$. No additional actions are needed along this branch to achieve the agent's goal.

So we see that the agent is able to determine at plan time that the above conditional plan achieves its goal.

Command	Precondition	Effects
ls -al z	$K(readable(z))$	**Plan Time:** $add(K_w, in\text{-}dir(x,z))$ $add(K_w, in\text{-}dir(x,z) \wedge readable(x))$ $add(K_w, in\text{-}dir(x,z) \wedge size(x) = y)$ **Run Time:** $exec(ls\ \text{-}al\ z, !file, !readable, !size)$ $add(K_f, in\text{-}dir(!file, z))$ $!readable \Rightarrow add(K_f, readable(!file))$ $add(K_f, size(!file) = !size)$ $add(LCW, in\text{-}dir(x,z))$ $add(LCW, in\text{-}dir(x,z) \wedge readable(x))$ $add(LCW, in\text{-}dir(x,z) \wedge size(x) = y)$
gzip x	$K(readable(x))$	**Plan Time:** $delete(K_v, size(x))$ **Run Time:** $exec(gzip\ x)$ $delete(K_f, size(x)), delete(K_v, size(x))$

Table 4: UNIX Domain Actions

At run time when the *stain* action is executed, the execution module determines if the colour of the stain is blue and binds the result to the run time variable *!stainblue*. The truth value of this variable will then determine whether or not *infected* or $\neg$*infected* is added to K_f. In either case, the plan executor will have sufficient information to correctly execute the rest of the conditional plan (cf. [Lev96]).

Notice that at plan time the agent is able to guarantee that the goal of curing the infection is achieved, by considering the possible consequences of the first action and planning appropriately. But, it is not until run time that the actual branch of the plan to execute in order to achieve the goal (either medicating or doing nothing) becomes known.

We close the paper with a final example taken the UNIX domain. The actions used in the example are given in Table 4.[11] This example uses a mechanism for posting exceptions to K_w and LCW information: specifying particular instances for which a K_w or LCW formula no longer holds. This mechanism will be explained in full in a later paper.

Say that in the real world we have *readable(1.ps)*, *readable(2.ps)*, *readable(old)*, *in-dir(1.ps, old)*, *size(1.ps)* $= 10,000$, and *in-dir(2.ps, new)*. The following conditional plan is intended to achieve the goal "If the file *1.ps* is in directory *old* and readable then compress it, and if *2.ps* is in directory *old* and readable compress it:" (1) *ls -al old*; (2) if *in-dir(1.ps, old)* and *readable(1.ps)* execute *gzip 1.ps*; (3) if *in-dir(2.ps, old)* and *readable(2.ps)* execute *gzip 2.ps*.

[11]We have simplified these UNIX actions somewhat for ease of presentation.

Say that the agent's initial knowledge state is $K_f = \{readable(1.ps), readable(2.ps), readable(old)\}$, with all of the other databases empty. Using the above action specifications we can project this conditional plan forward to determine what the agent's knowledge state would be at the various steps of the plan.

From the initial state we can conclude that the preconditions of *ls -al old* hold. Simulating this action we generate the new knowledge state where $K_w = \{in\text{-}dir(x, old), in\text{-}dir(x, old) \wedge readable(x), in\text{-}dir(x, old) \wedge size(x) = y\}$, and everything else is unaffected. From this knowledge state we have that $K_w(in\text{-}dir(1.ps, old))$, and $K(readable(1.ps))$. This entails that we know whether the branch condition of step 2 at this point in the plan, and hence the branch is legitimate.

Along the false branch we can conclude that $K(\neg(in\text{-}dir(1.ps, old))$ and $K(readable(1.ps))$, which is sufficient to show that the first goal is achieved on this branch. Along the true branch, K_f still contains *readable(1.ps)* which is sufficient to conclude that the preconditions of *gzip 1.ps* hold.

After simulating this action we obtain a new K_w in which the entry $in\text{-}dir(x, old) \wedge size(x) = y$ has been replaced by the entry $in\text{-}dir(x, old) \wedge size(x) = y \wedge (x \neq 1.ps)$ to reflect the fact that we no longer know the value of *size(1.ps)*. The mechanism that handles this update is part of an extension we have developed to deal with exceptions to K_w (and LCW) facts. This mechanism recognizes that the delete specified by *gzip* , $delete(K_v, size(1.ps))$, should not mean the simple removal of this item from the K_v database (in this case it is not even present in K_v). Rather, in this situation K_w allows us to conclude that we know this value, and so we must also update K_w. The mechanism we have developed posts exceptions to K_w and LCW facts. This allows us to update such facts without loosing excessive amounts of information (cf. [EGW97]).

The third step of the plan can be simulated in a similar manner to show that both of its branches also succeed in achieving the second goal (irrespective of the branch we took for step 2).

Turning now to execution time, the effects of the first and second steps of the plan are fairly straightforward. It is the third step that is interesting. At this stage of execution we would have executed the true branch of step 2 and would have $K_f = \{readable(\mathit{1.ps}), readable(\mathit{2.ps}), readable(old), in\text{-}dir(\mathit{1.ps}, old)\}$. At execution time a size fact for *1.ps* would have been added by step 1, but deleted by the execution of *gzip* . There are no facts in K_f about the file *2.ps* as it was not found to be in the listed directory, but we will have that $in\text{-}dir(x, old) \in LCW$. Now the inference algorithm can infer that $K(\neg(in\text{-}dir(\mathit{2.ps}, old))$, and the execution module can correctly realize that it should execute the false (null) branch of step 3's conditional.

References

[BK96] F. Bacchus and F. Kabanza. Using temporal logic to control search in a forward chaining planner. In M. Ghallab and A. Milani, editors, *New Directions in Planning*, pages 141–153. IOS Press, 1996.

[EGW94] O. Etzioni, K. Golden, and D. Weld. Tractable closed-world reasoning with updates. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, pages 178–189, 1994.

[EGW97] O. Etzioni, K. Golden, and D. Weld. Sound and efficient closed-world reasoning for planning. *Artificial Intelligence*, 1997. To appear, preprint available at ftp.cs.washington.edu.

[FHMV95] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge, Mass., 1995.

[Gar77] J. W. Garson. Quantification in modal logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, Vol. II*, pages 249–307. Reidel, Dordrecht, Netherlands, 1977.

[GEW94] K. Golden, O. Etzioni, and D. Weld. Omnipotence without omniscience: Efficient sensor management in planning. In *Proceedings of the AAAI National Conference*, pages 1048–1054, 1994.

[GW96] K. Golden and D. Weld. Representing sensing actions: The middle ground revisited. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, pages 174–185, 1996.

[Hin75] J. Hintikka. Impossible possible worlds vindicated. *Journal of Philosophical Logic*, 4:475–484, 1975.

[Lev96] H. Levesque. What is planning in the presense of sensing? In *Proceedings of the AAAI National Conference*, pages 1139–1146, 1996.

[McD96] D. McDermott. A heuristic estimator for means-end analysis in planning. In *Proceedings of the Third International Conference on A.I. Planning Systems*, 1996.

[Moo85] R. C. Moore. A formal theory of knowledge and action. In J. Hobbs and R. C. Moore, editors, *Formal Theories of the Commonsense World*, pages 319–358. Ablex Publishing Corp., Norwood, NJ, 1985.

[PG93] L. Pryor and Collins G. Cassandra: Planning for contingencies. Technical Report 41, Northwestern University, The Institute for the Learning Sciences, June 1993.

[PS92] M. Peot and D. Smith. Conditional nonlinear planning. In *Proceedings of the First International Conference on A.I. Planning Systems*, pages 189–197, 1992.

[SL93] R. B. Scherl and H. J. Levesque. The frame problem and knowledge-producing actions. In *Proceedings of the AAAI National Conference*, pages 689–695, 1993.

Reformulating Temporal Plans For Efficient Execution

Nicola Muscettola
Recom Technologies.
NASA Ames Research Center
Moffett Field, CA 94035
mus@ptolemy.arc.nasa.gov

Paul Morris
Caelum Research.
NASA Ames Research Center
Moffett Field, CA 94035
pmorris@ptolemy.arc.nasa.gov

Ioannis Tsamardinos
Intelligent Systems Program
University of Pittsburgh
Pittsburgh, PA 15260
tsamard@cs.pitt.edu

Abstract

The Simple Temporal Network formalism permits significant flexibility in specifying the occurrence time of events in temporal plans. However, to retain this flexibility during execution, there is a need to propagate the actual execution times of past events so that the occurrence windows of future events are adjusted appropriately. Unfortunately, this may run afoul of tight real-time control requirements that dictate extreme efficiency. The performance may be improved by restricting the propagation. However, a fast, locally propagating, execution controller may incorrectly execute a consistent plan. To resolve this dilemma, we identify a class of *dispatchable* networks that are guaranteed to execute correctly under local propagation. We show that every consistent temporal plan can be reformulated as an equivalent dispatchable network, and we present an algorithm that constructs such a network. Moreover, the constructed network is shown to have a minimum number of edges among all such networks. This algorithm will be flown on an autonomous spacecraft as part of the Deep Space 1 Remote Agent experiment.

1 Introduction

When designing and implementing control systems operating in a physical world it is important to correctly deal with the metric nature of time. For example, deadlines are typically upper bounds on the value of the occurrence time of certain events (e.g., end of a task). The control system can guarantee a correct execution only if specified time constraints are satisfied for any possible execution. We are interested in the class of high-level control architectures that distinguish between a *deliberative* layer, or *planner*, and a *reactive* layer, or *executive* [10, 1, 13, 6, 11, 9, 2].

This paper is concerned with the properties that must be satisfied by temporally flexible plans in order to be correctly executed by a simplified, fast execution algorithm. The speed of an execution algorithm is central to ensuring that a plan can be robustly executed in real-time, a condition of crucial importance in mission critical applications such as autonomous spacecraft operations [10] and avionics control systems [3]. Unlike fixed time schedules, temporally flexible plans allow an executive to seamlessly adjust to delays and fluctuations of task durations. However, the cost of this flexibility is that the executive must constantly adjust the plan during execution by performing some amount of constraint propagation. The time spent doing this propagation adds to the total time needed to start or end any task in the plan. The latter time is equivalent to the intrinsic time uncertainty on the exact time of occurrence of any event in the plan [7, 8]. The more precise we want the execution of a plan to be, the less propagation an execution algorithm should perform. In this paper, we precisely define fast execution by giving a simple controller algorithm and we describe *dispatchability*, a formal property that identifies whether a plan is amenable to fast execution or not. We also discuss how a non-dispatchable plan can be transformed in polynomial time into a dispatchable plan, and we show that the resulting plan has the desirable property of being minimal in the number of edges among all dispatchable plans.

2 Dispatching Executions

The type of plan that we are interested in is a *temporal plan*, i.e., a partial order of tasks with metric time information. We refer to the start and end times of a task as two separate *events* or *timepoints*. A plan satisfies the following conditions: (1) for each task, the start and end events must be separated by a non-negative duration $[d, D]$; (2) additional separation constraints $[s, S]$ may be specified between the start and/or the end of any two tasks. A temporal bound-constraint $[b, B]$ (either duration or separation) from an event A to an event B constrains the possible values of the times of occurrence of A and B, T_A and T_B respectively, such that $b \leq T_B - T_A \leq B$. We assume that the plan contains no disjunctive bound-constraints between two events, i.e., the graph of events and bound-constraints is a Simple Temporal Network (STN) in the sense of Dechter, Meiri and Pearl [5]. Without loss of generality, we also assume that the STN graph is connected.

We concentrate on the process through which the executive selects individual events and executes them, i.e., assigns to them a specific time of occurrence that is consistent with the overall plan. It has been established [5] that finding the ranges of execution times for each event, the event's *time bounds*, is equivalent to solving two single-source shortest-path problems [4] on a simple transformation of the STN graph. Furthermore, if the STN is consistent, then for each event A it is possible to arbitrarily pick a time T_A within its time bounds and find corresponding times for the other events such that the set of occurrence times for all events satisfies the plan constraints. This suggests a "naive" execution algorithm that iteratively: (1) selects an event such that the current time is within the event's time bound and the event is *enabled*, i.e., all events that must directly precede it in the STN have been executed; (2) assigns the current time to the event; and (3) propagates the consequences of "collapsing" the event's time bounds to every other time bound. The iteration continues until all events have been executed.

There are two problems with the naive execution algorithm. The first is that precisely estimating the propagation cost for a general STN is difficult and may require considering the possible propagations in a large number of possible execution conditions. Without such analysis, the best we can do is to give a bound that depends on the total size of the plan; more precisely the bound corresponds to running the Dijkstra algorithm[1] twice on the graph. The complexity of this propagation is $O(e + n \log n)$, where e is the total number of edges and n the total number of nodes in the STN. A second, more serious problem is that selecting events on the sole basis of time bound information

[1]Since the STN is guaranteed to remain consistent, it is possible to avoid using the more costly Bellman-Ford-Moore algorithm.

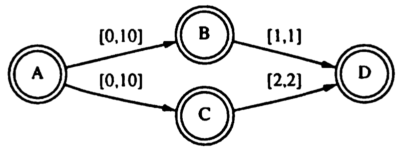


Figure 1: Simple Temporal Network.

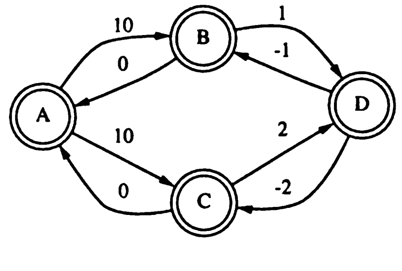


Figure 2: Distance Graph.

and precedence enablement may lead to incorrect executions. Consider the example network in Figure 1. Intuitively this network corresponds to two tasks BD and CD of fixed durations, respectively 1 and 2 time units, that synchronize at the end (event D) and must start within 10 time units of a time origin (event A). Figure 2 shows the corresponding *distance graph* [5], suitable for shortest-path propagation. If we assume that event A always occurs at time 0, events B and C will initially obtain time bounds $\langle 1, 10 \rangle$ and $\langle 0, 9 \rangle$ respectively. (Recall from [5] that the lower bound for a node is computed as the negation of the shortest-path distance from the node to the time origin, while the upper bound is simply the shortest-path distance from the origin to the node.) Suppose now that the current time is 5, but tasks BD and CD have not yet started. Since the time bounds of both B and C contain time 5, the naive execution algorithm may very well select for execution event B only to discover after propagation that event C should have started at time 4 in order for the plan to execute consistently. Thus, the naive approach does not guarantee correct execution of a plan under all execution conditions.

The problem with the plan in Figure 1 is that there is an implicit synchronization constraint that requires C to be executed exactly 1 time unit before B. When the execution reaches B and C, this implicit constraint can only be detected by increasing the lower bound of waiting events to the current time, and propagating, *before* considering which event to select. Although this fixes the consistency problem, it does not improve the real-time performance. Indeed, it makes it worse since

```
TIME DISPATCHING ALGORITHM:
    1. Let
         A = {start_time_point}
         current_time = 0
         S = {}
    2. Arbitrarily pick a time point TP in A such
       that current_time belongs to TP's time bound;
    3. Set TP's execution time to current_time and add
       TP to S;
    4. Propagate the time of execution
       to its IMMEDIATE NEIGHBORS in the distance
       graph;
    5. Put in A all time points TPx such that all
       negative edges starting from TPx have a
       destination that is already in S;
    6. Wait until current_time has advanced to
       some time between
          min{lower_bound(TP) : TP in A}
       and
          min{upper_bound(TP) : TP in A}
    7. Go to 2 until every time point is in S.
```

Figure 3: The Dispatching Execution Controller.

we may now have to propagate the new lower bound from several waiting events rather than from a single selected event.

On the other hand, to fix the performance problem, we would like to restrict the execution algorithm to use a *local propagation* that, on the basis of the execution time of an event, adjusts only the time bounds of the *neighboring* events. However, this makes the consistency problem worse. To see this, note that with the plan in Figure 1, even if C is executed first, local propagation would allow B to be executed *more* than one time unit after C, which also violates the implicit constraint.

Notwithstanding these considerations, there are many networks that *are* successfully executed by using the naive execution algorithm. This continues to be true even when the propagation is restricted to be local. In fact, as we will see, every consistent STN is equivalent to such a network.

Figure 3 shows a detailed local propagation algorithm that we call the *dispatching execution controller*. Note that the flexible wait in step 6 provides some room for responding to unmodeled external contingencies. This can include unexpected events (in contrast to work that deals with anticipated uncontrollable events [12]). Step 5 is a precise formulation of the *enablement* requirement that prevents execution of a node until all its enabling nodes have first been executed. Note that with this formulation a deadlock situation cannot occur, since in a consistent distance graph there are no negative cycles. An execution carried out by the dispatching execution controller is called a *dispatching execution*. An STN is said to be *dispatchable* if it is always correctly executed by the dispatching execution controller.

The propagation time needed to execute a dispatchable plan is easy to estimate, and varies directly with b, the maximum number of edges that can enter or exit an event in the associated distance graph.

In this paper, it is shown that every consistent STN can be reformulated as an equivalent dispatchable network. This is achieved by (1) constructing the all-pairs shortest path network (which is shown to be dispatchable), and (2) eliminating unneeded edges to obtain an equivalent dispatchable network of minimum size.

3 Finding dispatchable networks

We will use the following notation with respect to distance graphs. Given a timepoint X, the expression T_X denotes its execution time with respect to some schedule or execution. If X and Y are two timepoints, XY denotes an edge from X to Y, and $b(X,Y)$ is its distance or length. (The edge XY represents the constraint $T_Y - T_X \leq b(X,Y)$.) We write $|XY|$ to denote the distance along a shortest path from X to Y, or ∞ if no path exists. (Note that $|XY|$ may be negative in distance graphs.) The proofs of the theorems (and supporting lemmas) are contained in the Appendix.

The first result is useful for simplifying the local propagation required in a dispatching execution. Recall [5] that in an STN distance graph, the upper bounds of timepoints are propagated in the forward direction of edges, whereas lower bounds are transmitted in the reverse direction.

Theorem 1 *In a dispatching execution, upper-bound propagations along negative edges, and lower-bound propagations along non-negative edges, are both ineffectual, i.e., they do not affect the course of the execution.*

Note that theorem 1 shows that, in a dispatching execution, the upper and lower-bound propagations can be confined to disjoint sets of edges.

In the remainder of the paper, unless stated to the contrary, it is convenient to use the term execution to mean dispatching execution.

We next investigate what is needed to obtain dispatchable networks. Recall that any STN can be rewritten as an All-Pairs shortest-path network (called the *d-graph* in [5]). We have the following.

Theorem 2 *Every All-Pairs shortest-path network is dispatchable.*

Although the All-Pairs network is dispatchable, it has some obvious disadvantages. In particular, the propagation at each node requires time proportional to n, the number of nodes, in *every* case. Fortunately, we can do better. Relying on Theorem 2, we will adopt the following strategy. Given an arbitrary STN, we first construct the equivalent All-Pairs network. Then we strip out unneeded edges, the goal being to end up with an equivalent dispatchable network of manageable size. Although it is possible to construct examples where there are no unneeded edges, experiments show that, typically, a minimal dispatchable network is found that is of size comparable to that of the original network.

To make this work, we need some means of identifying unneeded edges. Formally, an edge is *unneeded* if its removal does not admit any new executions. Intuitively, this condition is satisfied if its propagations are always superseded by those of some other edge. By Theorem 1, the only cases we need to consider are forward propagations along non-negative edges, which may affect upper-bounds, and backward propagations along negative edges, which may affect lower-bounds.

Recall that for an edge XY, the expression $T_X + b(X,Y)$ constitutes the upper-bound value propagated forward from X to Y, while $T_Y - b(X,Y)$ is the lower-bound value propagated backwards from Y to X.

This leads to the following definition.

Definition 1 *(a) Consider two edges AC and BC with the same destination C, and suppose the lengths of both are non-negative. We say BC* upper-dominates *AC if in every consistent execution, $T_B + b(B,C) \leq T_A + b(A,C)$.*

(b) Consider two edges AC and AB with the same source A, and suppose the lengths of both are negative. We say AB lower-dominates *AC if in every consistent execution, $T_B - b(A,B) \geq T_C - b(A,C)$*

(c) Finally, we say an edge E1 dominates *an edge E2 if either E1 upper-dominates E2 or E1 lower-dominates E2.*

The following results pertain to graphs that satisfy the triangle inequality. Note that these include the All-Pairs graph and subgraphs derived from it by removing edges.

The next theorem provides a characterization of the dominance relation that is more easily checked by an algorithm.

Theorem 3 (Triangle Rule) *Consider a consistent STN where the associated distance graph satisfies the triangle inequality.*

(1) A non-negative edge AC is upper-dominated by another non-negative edge BC if and only if $|AB| + |BC| = |AC|$.

(2) A negative edge AC is lower-dominated by another negative edge AB if and only if $|AB| + |BC| = |AC|$.

We next consider the removal of edges. We are interested in knowing whether this allows a new execution that *deviates* or differs from those that were possible before the removal. We will say an edge is *unneeded* if its removal does not produce a new dispatching execution. In this case, removing the edge will not introduce an incorrect execution that did not exist before.

The following result confirms our interest in the dominance relation.

Theorem 4 (Filtering Theorem) *An edge in a dispatchable network that satisfies the triangle inequality is unneeded if and only if it is dominated by some other edge.*

Theorems 3 and 4 together allow us to remove an edge AC from the All-Pairs shortest-path network if there is another node B such that $|AB| + |BC| = |AC|$, and either both $|AC|$ and $|AB|$ are negative, or both $|AC|$ and $|BC|$ are non-negative. In the former case, AC is lower-dominated by AB, while in the latter, it is upper-dominated by BC.

Notice that since the removal of a dominated edge leaves the set of executions unchanged, it does not interfere with the dominance relation between other pairs of edges. This suggests a potential for multiple removals, where the triangle rule conditions can conveniently be checked in the fixed All-Pairs network. However, some interaction is still possible where edges dominate each other; obviously, only one may be removed on account of the other (although they may both be removed if dominated by a third edge). To see how edge removals may be combined, we consider further properties of the dominance relation.

Theorem 5 *The dominance relation is reflexive and transitive.*

A binary relation that is reflexive and transitive is called a *preorder*. It is well-known that a preorder $\leq$ induces an equivalence relation $\equiv$, defined by $x \equiv y$ if $x \leq y$ and $y \leq x$. Moreover, the equivalence classes are partially ordered by the $\leq$ relation.

In the case of the dominance relation, the induced equivalence classes will be useful in formulating a multiple removal strategy, as discussed in the next section.

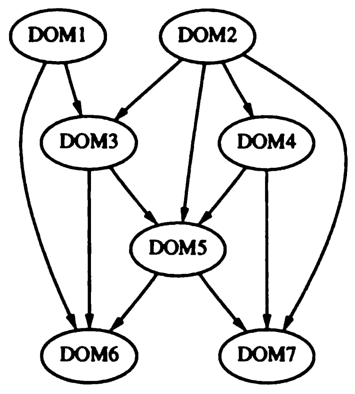


Figure 4: Domination Partial Order.

4 Minimality of filtered network

From the properties of the DOMINATES relations (Definition 1) we see that an all-pair shortest path network can give rise to a number of "minimal" dispatchable networks, where minimality means that the filtered network has a minimal number of edges. (Not to be confused with the "tightness" minimality property defined in [5].) We now wish to show that all of these have the same number of edges.

An example of a partial order structure induced by DOMINATES is shown in Figure 4.

Each set DOMi corresponds to an equivalence class for the DOMINATES relation. All links in such a class dominate each other symmetrically. The link from one equivalence class to another (e.g., from DOM3 and DOM5) represents the fact that any node in the first class dominates all of the nodes in the second class. This property follows straightforwardly from the transitivity of the DOMINATES relation. A minimal number of globally dominating edges can be obtained by picking one bound per each "source" DOMj equivalence class (in the case in Figure 4, DOM1 and DOM2). Since this selection can be done arbitrarily for each "source" DOMi, in general there is a potentially very large number of different minimal dispatchable networks obtainable from an all-pair shortest path network. However, from the point of view of the execution controller, all of the networks are equivalent, and they all have the same number of edges, so generating any one of them is sufficient.

5 Edge filtering algorithm

In this section we describe an algorithm that generates one of the minimal networks. First we describe the algorithm and then we prove its correctness and

```
procedure MARK-EDGES-FOR-ELIMINATION
begin
  for each pair of intersecting edges
  do begin
    if both dominate each other
    then
        if neither is marked
        then
            Pick one arbitrarily
            and mark it
        else
            Do nothing
    else if one dominates the other
    then
        Mark the dominated edge
  end
end
```

Figure 5: Minimal dispatch filtering algorithm.

minimality.

Input: A consistent all-pair shortest path graph $< N, b(.,.) >$ where N is a set of time points with cardinality n and $b(.,.)$ is a total function $N \times N \rightarrow \Re$ such that $b(X,Y)$ is the length of the shortest path link from X to Y.

Output: A consistent minimal dispatchable network $< N, b'(.,.) >$, where $b'(.,.)$ is a restriction of $b(.,.)$ to a subset of $N \times N$.

The central routine in the algorithm is shown in figure 5. The routine visits edges in the network, marking some of them for elimination. A subsequent routine deletes the marked edges. The dominance relations can be established by applying the Triangle Rule of Theorem 3.

In the marking algorithm, two edges *intersect* if they either have the same source or the same destination. As a matter of implementation, all the pairs of intersecting edges can be conveniently visited by iterating over each set of three vertices or *triangle* and considering the edges between them.

It remains only to show that, with respect to the dominance partial order, as depicted in Figure 4, the application of the marking algorithm will mark all edges in the "non-source" equivalence classes (ones that have a predecessor class), and will eliminate all but one edge in the "source" equivalence classes.

First, consider an edge belonging to a "non-source" equivalence class DOMi. Eventually, it will be tested against an edge in an equivalence class DOMj that precedes DOMi, and at that time it will be marked for elimination.

Next we treat the case of a "source" equivalence class DOMk. Consider the last time that the marking algorithm is applied to a pair of edges in DOMk that are both unmarked. Only one of the two edges will survive. The one that survives will survive until the end, since all other applications of the marking algorithm to pairs of edges in DOMk will necessarily involve at least one marked edge, which will prevent any additional marking from occurring. Furthermore, there must be exactly one edge left. Suppose to the contrary that there are at least two survivors E1 and E2 in DOMk. At some point the algorithm would have considered E1 and E2 as a pair, and if both were unmarked, it would have marked one of them. Thus, both could not have survived. It follows that only one edge per source equivalence class will survive after the termination of the algorithm.

Observe that when two unmarked edges dominate each other, there is a choice of which to eliminate. Thus, there are many equivalent minimal graphs that could be produced by the algorithm. Notice, however, they all contain exactly one edge from each of the "source" equivalence classes, and so they all have the same number of edges. This shows the algorithm is "best possible," in the sense that it produces a graph with a globally minimum number of edges.

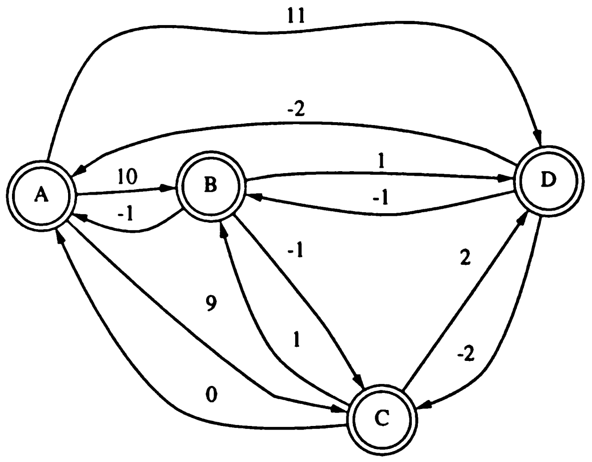


Figure 6: All Pairs Graph.

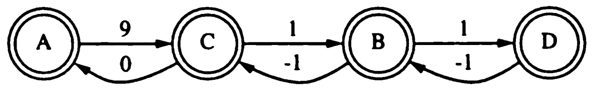


Figure 7: Final Filtered Graph.

6 Example and Experimental results

Continuing the example started with Figure 1, Figure 6 represents the fully connected distance graph obtained after application of the all-pairs shortest-path propagation. (Note that the BA and AC distances have decreased from the edge values in the original distance graph due to alternate, shorter paths.) After the application of the filtering algorithm described in section 5 we obtain the minimal dispatchable graph in figure 7. Notice that the final STN contains one less time-bound edge than the starting network in figure 1.

The algorithm described in Section 5 already has a practical application. It is being used in the New Millennium Remote Agent [10], a control architecture that will operate autonomously the Deep Space 1 (DS1) spacecraft in a 6 day experiment scheduled for October 1998. Table 1 summarizes the experimental results on the three plans that will be nominally generated and executed during the experiment.

All the results refer to distance graphs like those in figure 2, figure 6 and figure 7. The results show that the minimal dispatchable network is significantly smaller than the all-pairs dispatchable network, having between 5% and 7% the number of edges of the all-pairs network. They also show that the size of the minimal dispatchable network is smaller than the original plans generated by the on-board planner, having between 40% and 70% the edges of the original plan. Notice that even if the original plan were dispatchable, the minimal plans improve the real-time guarantee (proportional to the maximum number of branching edges at a node) between 1.5 and 2.2 times with respect to the original plan.

A Proofs

Lemma 1 *Given any consistent schedule for any STN, there is a dispatching execution that realizes the schedule.*

Proof: First we show that the enablement restriction does not exclude any consistent schedules. To see this, note that for any link $X \rightarrow Y$, we have $T_Y - T_X \leq b(X,Y)$, and so $T_Y < T_X$ if $b(X,Y)$ is negative. Second, note that the restriction to local propagation is actually more lenient in terms of nar-

Table 1: Minimal Dispatchability For DS1 Plans

	nodes	original edges	All-Pair edges	minimal edges	original max. branch	minimal max. branch
PLAN-1	56	390	3080	156	18	11
PLAN-2	39	144	1482	102	14	9
PLAN-3	66	424	4290	192	26	12

rowing the time bounds, so all consistent choices for execution time remain. □

In light of lemma 1, we may use the terms "consistent schedule" and "consistent execution" interchangeably in the subsequent proofs.

For the next result, recall that in STN propagation, the upper bounds of timepoints are propagated in the forward direction of edges, whereas lower bounds are transmitted in the reverse direction.

Theorem 1 *In a dispatching execution, upper-bound propagations along negative edges, and lower-bound propagations along non-negative edges, are both ineffectual, i.e., they do not affect the course of the execution.*

Proof: First we remark that a propagation to an already executed node is always ineffectual, since it cannot narrow the bounds further. Note also that such a propagation cannot generate an inconsistency, since the constraint has already been enforced by a prior propagation in the reverse direction. Now consider an upper-bound propagation along an edge $X \to Y$ that has a negative length. Because of the enablement condition, Y must have been executed before X. Thus, by the remark above, the propagation is ineffectual. Next consider a lower-bound propagation along a reverse edge $X \leftarrow Y$ that has non-negative length w. If Y has been executed before X, then we are done by the earlier remark. Otherwise, Y must occur at or after X. In that case, the edge constraint requires that $T_X - T_Y \leq w$, which can be rewritten as $T_Y \geq T_X - w$. Since Y is not occurring before X anyway, this bound does not constrain Y, and is subsumed by the actual execution time of Y. □

In the remainder of the proofs, it is convenient to use the term *execution* to mean *dispatching execution*.

Theorem 2 *Every All-Pairs shortest-path network is dispatchable.*

Proof: First we show that a full-propagating execution controller that respects the enablement conditions cannot generate an inconsistency. The theory of Simple Temporal Networks [5] guarantees that any locally consistent assignment can be extended to a global one. This means that (full) propagation during execution will not reduce any timepoint's bounds to the empty set. Thus, the only possibility for incorrect execution is if a pending unexecuted timepoint X is forced into the past by a propagation. For this to happen, there must be a shortest path of negative distance from some currently executing timepoint Y to X. In this case, because of the All-Pairs property, there will be a single edge from Y to X that has a negative length. But then the enablement condition would have forced X to be executed before Y, giving a contradiction.

Next, we observe that local propagation in the All-Pairs shortest-path network simulates full-propagation. It follows that the dispatching execution controller will not generate an inconsistency. Thus the All-Pairs network is dispatchable. □

Lemma 2 *Let A and B be timepoints in a consistent STN. Then in all consistent schedules, $T_B - T_A \leq |AB|$. Moreover, if $|AB|$ is finite, there is at least one consistent schedule where $T_B - T_A = |AB|$. If $|AB|$ is infinite, there are consistent schedules in which $T_B - T_A$ is arbitrarily large.*

Proof: The first part is immediate upon summing the inequalities for each edge in a shortest path. To see the second part, consider adding a link from B to A with length $-|AB|$. The network must still be consistent, since the shortest cycle through the edge BA has length $-|AB| + |AB| = 0$. Thus, there is at least one consistent schedule for the new network. This satisfies $T_A - T_B \leq -|AB|$ Combining this with the inequality of the first part gives $T_B - T_A = |AB|$. The result then follows, since this is also a consistent schedule for the original network. A similar method works for the infinite case by adding a negative link whose absolute value is arbitrarily large. □

Theorem 3 *Consider a consistent STN where the associated distance graph satisfies the triangle inequality.*

(1) A non-negative edge AC is upper-dominated by another non-negative edge BC if and only if $|AB| +$

$|BC| = |AC|$.

(2) A negative edge AC is lower-dominated by another negative edge AB if and only if $|AB| + |BC| = |AC|$.

Proof: First consider edges AC and BC in (1). Suppose $|AB|+|BC| = |AC|$. By the first part of lemma 2, in any consistent schedule, we have $T_B - T_A \leq |AB|$. It follows that $T_B + |BC| \leq T_A + |AB| + |BC|$. Thus, $T_B + |BC| \leq T_A + |AC|$ by our hypothesis. We can deduce from the triangle inequality that $|BC| = b(B,C)$ and $|AC| = b(A,C)$, so BC upper-dominates AC. Conversely, suppose that BC upper-dominates AC. Then in every consistent schedule, $T_B + |BC| \leq T_A + |AC|$. If $|AB| + |BC| \neq |AC|$, the only option allowed by the triangle inequality is $|AC| < |AB|+|BC|$. Combining the inequalities yields that $T_B - T_A < |AB|$ in every consistent schedule. But this contradicts the second part of lemma 2 if $|AB|$ is finite. In the infinite case, we can choose some finite value K such that $|AC| < K + |BC|$. Then $T_B - T_A < K$, which also contradicts lemma 2.

Now consider edges AC and AB in (2). Suppose again $|AB| + |BC| = |AC|$. The first part of lemma 2 gives us $T_C - T_B \leq |BC|$. Thus, $T_C + |AB| \leq T_B + |AB| + |BC| = T_B + |AC|$. This can be rewritten as $T_B - |AB| \geq T_C - |AC|$, and the result follows using the triangle inequality on $|AB|$ and $|AC|$. Conversely, if AB lower-dominates AC, then $T_B - |AB| \geq T_C - |AC|$ in every consistent schedule. Suppose $|AC| < |AB| + |BC|$. Then $T_C - T_B < C < |BC|$ for some C in every consistent schedule, again contradicting lemma 2. Thus, $|AC| = |AB| + |BC|$. □

For the following lemma, a *trace* of an execution or partial execution is a sequence $S_0, S_1, \ldots$ where S_i is the set of events that are executed at time i. A *deviation* occurs in a partial execution after an edge removal when the trace is no longer a prefix of any of the traces before the edge removal. This may happen either because of the addition or omission of an event in the trace.

Lemma 3 *Suppose the removal of an edge AC from a dispatchable network produces a new, deviant execution. If AC is a negative edge, then the earliest time at which the execution deviates is when A is executed. Otherwise it is when C fails to be executed within $b(A,C)$ time units after A. In either case, the AC constraint is violated.*

Proof: The following terminology will be useful. The *background state* of any time-point during an execution refers to its current time-bounds and enablement/execution status.

Consider first the case where AC is a negative edge. Then AC is an enablement edge for A. Let T be the time of first deviation in the schedule. Note that T cannot occur before A is executed, since before then the only differences in the background states of the timepoints are in the more lenient enablement status and lower bound of A, which only allow it to be executed sooner.

Now suppose the AC constraint is not violated. Then C must have been executed before A, and after C and A are executed, the background state of every timepoint will mirror that before the removal of AC. In that case, the execution will not deviate at or after the execution of A. Thus, the AC constraint must be violated. It follows that T cannot occur after A is executed, and so it must occur precisely when A is executed, and either C was not executed before A, or A occurs too soon after C.

The case where AC is non-negative is similar. Again let T be the time of first deviation. Note that before A is executed, the background state of every node (including A, since propagations from C to A are ineffectual) is the same as before the edge removal. Thus, T cannot occur before A is executed. After A is executed, the only difference in the background states is in the upper bound of C, which may be larger than before the edge removal. Again, if the AC constraint is satisfied, the execution will not deviate. Thus, the AC constraint must be violated, and the deviation occurs when C fails to execute within the allotted time after A. □

Theorem 4 *An edge in a dispatchable network that satisfies the triangle inequality is unneeded (in the sense that its removal does not alter the set of executions) if and only if it is dominated by some other edge.*

Proof: Suppose an edge AC is dominated by another edge. We will show the set of dispatching executions is unaltered after the dominated edge is removed. Consider first the easier case where a negative edge AC is lower-dominated by another negative edge AB. Suppose that after AC is removed, there is a new, deviant execution. By lemma 3, the deviation first occurs when A is executed. Note that B is still an enablement condition for A (AB was not removed), so B is executed before A. Since $T_B - b(A,B) \geq T_A - b(A,C)$, the propagation from B to A subsumes that from C to A by the time A is executed. This implies the execution time of A does not deviate, which is a contradiction.

Next consider the situation where a non-negative edge AC is upper-dominated by another non-negative edge

BC. Suppose that after AC is removed, there is a new, deviant execution. By lemma 3, AC is violated in the deviant execution, so A must be executed before C, and the deviation occurs $|AC|$ time units after A when C fails to be executed. (By the triangle inequality $b(A, C) = |AC|$.) By theorem 3, $|AC| = |AB| + |BC|$. Since $|BC| \geq 0$, it follows that $|AC| \geq |AB|$. Thus, B must be executed before the deviation occurs. Then by the dominance condition, the propagation from B to C subsumes that from A to C, implying that C does not deviate.

Conversely, suppose an edge AC is not dominated. Consider first the case in which AC is non-negative. By theorem 3, for any node B we have either BC is negative or $|AB| + |BC| > |AC|$. In the former case, any propagation from B to C is ineffectual by theorem 1. With respect to the latter cases, consider a propagation of distances bounds with A as the source. The upper bounds for each node will then form a consistent schedule. This shows that there is some execution in which $T_B = T_A + |AB|$ for every B such that $|AB| + |BC| > |AC|$. It follows that $T_B + |BC| > T_A + |AC|$ in this execution, for every such B. This means that if the edge AC is removed, the execution can be modified to delay the execution time of C until more than $|AC|$ time units after A, which is a deviation. Thus, AC is needed. The proof when AC is negative is similar. □

Theorem 5 *The dominance relation is reflexive and transitive.*

Proof: Reflexivity is immediate from the definitions. Transitivity follows from the fact that only dominances with edges of the same sign can be chained. □

Acknowledgments

We thank Pandu Nayak, David Smith, and the referees for suggestions to improve the presentation of these results.

References

[1] R. P. Bonasso, D. Kortenkamp, D. Miller, and M. Slack. Experiences with an architecture for intelligent, reactive agents. *JETAI*, 9(1), 1997.

[2] J. Bresina, M. Drummond, , and S. Kedar. *Reactive, Integrated Systems Pose New Problems for Machine Learning.* Morgan Kaufmann, San Mateo, California, 1993.

[3] T. Carpenter, K. Driscoll, and K. Carciofini J. Hoyme. Arinc 659 scheduling: Problem definition. In *Proceedings of 1994 IEEE Real Time System Symposium.* IEEE, 1994.

[4] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms.* MIT press, Cambridge, MA, 1990.

[5] R. Dechter, I Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, May 1991.

[6] Brian Drabble, Austin Tate, and Jeff Dalton. O-plan project evaluation experiments and results. Oplan Technical Report ARPA-RL/O-Plan/TR/23 Version 1, AIAI, July 1996.

[7] N. Muscettola, P. Morris, B. Pell, and B. Smith. Issues in temporal reasoning for autonomous control systems. In F. Anger, editor, *Working Notes from the 1997 AAAI workshop on Spatial and Temporal Reasoning*, 1997. available at http://ic-www.arc.nasa.gov/ic/projects/Executive/papers/aaai97-temporal.ps.

[8] N. Muscettola and B. Pell. Real-time execution of temporal plans. Technical Report in preparation, Computational Sciences Division, NASA Ames Research, 1997.

[9] David Musliner, Ed Durfee, and Kang Shin. Circa: A cooperative, intelligent, real-time control architecture. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(6), 1993.

[10] B. Pell, D. E. Bernard, S.A. Chien, E. Gat, N. Muscettola, P. P. Nayak, M.D. Wagner, and B.C. Williams. An autonomous spacecraft agent prototype. *Autonomous Robotics*, forthcoming, 1997.

[11] Reid Simmons. An architecture for coordinating planning, sensing, and action. In *Procs. DARPA Workshop on Innovative Approaches to Planning, Scheduling and Control*, pages 292–297, San Mateo, CA, 1990. DARPA, Morgan Kaufmann.

[12] T. Vidal and M. Ghallab. Dealing with uncertain durations in temporal constraint networks dedicated to planning. In *Proc. of 12th European Conference on Artificial Intelligence (ECAI-96)*, pages 48–52, 1996.

[13] D. E. Wilkins, K. L. Myers, J. D. Lowrance, and L. P. Wesley. Planning and reacting in uncertain and dynamic environments. *JETAI*, 7(1):197–227, 1995.

Execution Monitoring of High-Level Robot Programs

Giuseppe De Giacomo[*]
Dip. di Informatica e Sistemistica
Università di Roma "La Sapienza"
Roma, I-00198, Italy
degiacomo@dis.uniroma1.it

Ray Reiter
Dept. of Computer Science
University of Toronto
Toronto, M5S 3G4, Canada
reiter@cs.toronto.edu

Mikhail Soutchanski
Dept. of Computer Science
University of Toronto
Toronto, M5S 3G4, Canada
mes@cs.toronto.edu

Abstract

Imagine a robot that is executing a program on-line, and, insofar as it is reasonable to do so, it wishes to continue with this on-line program execution, no matter what exogenous events occur in the world. *Execution monitoring* is the robot's process of observing the world for discrepancies between the actual world and its internal representation of it, and recovering from such discrepancies.

We provide a situation calculus-based account of such on-line program executions, with monitoring. This account relies on a specification for a single-step interpreter for the logic programming language *Golog* . The theory is supported by an implementation that is illustrated by a standard blocks world in which a robot is executing a *Golog* program to build a suitable tower. The monitor makes use of a simple kind of planner for recovering from malicious exogenous actions performed by another agent. After performing the sequence of actions generated by the recovery procedure, the robot eliminates the discrepancy and resumes executing its tower-building program.

We also indicate how, within the formalism, one can formulate various correctness properties for monitored systems.

1 Introduction and motivation.

Imagine a robot that is executing a program on-line, and, insofar as it is reasonable to do so, it wishes to continue with this on-line program execution, no matter what exogenous events occur in the world. An example of this setting, which we treat in this paper, is a robot executing a program to build certain towers of blocks in an environment inhabited by a (sometimes) malicious agent who might arbitrarily move some block when the robot is not looking. The robot is equipped with sensors, so it can observe when the world fails to conform to its internal representation of what the world would be like in the absence of malicious agents. What could the robot do when it observes such a discrepancy between the actual world and its model of the world? There are (at least) three possibilities:

1. It can give up trying to complete the execution of its program.
2. It can call on its programmer to give it a more sophisticated program, one that anticipates all possible discrepancies between the actual world and its internal model, and that additionally instructs it what to do to recover from such failures.
3. It can have available to it a repertoire of *general* failure recovery methods, and invoke these as needed. One such recovery technique involves planning; whenever it detects a discrepancy, the robot computes a plan that, when executed, will restore the state of the world to what it would have been had the exogenous action not occurred. Then it executes the plan, after which it resumes execution of its program.

Execution monitoring is the robot's process of observing the world for discrepancies between "physical reality", and its "mental reality", and recovering from such perceived discrepancies. The approach to execution monitoring that we take in this paper is option 3 above. While option 2 certainly is valuable and important, we believe that it will be difficult to write programs that take into account all possible exceptional cases. It will be easier (especially for inexperienced programmers) to write simple programs in a language like *Golog* , and have a sophisticated execution monitor (written by a different, presumably more experienced programmer) keep the robot on track in its actual execution of its program.

In general, we have the following picture: The robot is executing a program on-line. By this, we mean that it is physically performing actions in sequence, as these

[*]Author names are alphabetical.

are specified by the program.[1] After each execution of a primitive action or of a program test action, the execution monitor observes whether an exogenous action has occurred. If so, the monitor determines whether the exogenous action can affect the successful outcome of its on-line execution. If not, it simply continues with this execution. Otherwise, there is a serious discrepancy between what the robot sensed and its internal world model. Because this discrepancy will interfere with the further execution of the robot's program, the monitor needs to determine corrective action in the form of another program that the robot should continue executing on-line instead of its original program. So we will understand an execution monitor as a mechanism that gets output from sensors, compares sensor measurements with its internal model and, if necessary, produces a new program whose on-line execution will make things right again.

Our purpose in this paper is to provide a situation calculus-based account of such on-line program executions, with monitoring. To illustrate the theory and implementation, we consider a standard blocks world as an environment in which a robot is executing a *Golog* program to build a suitable tower. The monitor makes use of a simple kind of planner for recovering from malicious exogenous actions performed by another agent. After the robot performs the sequence of actions generated by the recovery procedure, the discrepancy is eliminated and the robot can resume building its goal tower.

2 The Situation Calculus and Golog

The version of the situation calculus that we use here has been described in [18], [22], and elsewhere. The situation calculus is a second order language specifically designed for representing dynamically changing worlds. All changes to the world are the result of named *actions*. A possible world history, which is simply a sequence of actions, is represented by a first order term called a *situation*. The constant S_0 is used to denote the *initial situation*, namely the empty history. Non-empty histories are constructed using a distinguished binary function symbol *do*; $do(\alpha, s)$ denotes the successor situation to s resulting from performing the action α. Actions may be parameterized. For example, $put(x, y)$ might stand for the action of putting object x on object y, in which case $do(put(A, B), s)$ denotes that situation resulting from placing A on B when the history is s. In the situation calculus, actions are denoted by first order terms, and situations (world histories) are also first order terms. For example, $do(putdown(A), do(walk(L), do(pickup(A), S_0)))$ is the situation denoting the world history consisting of the sequence of actions [pickup(A), walk(L), putdown(A)]. Notice that the sequence of actions in a history, in the order in which they occur, is obtained from a situation term by reading off the actions from right to left.

Relations whose truth values vary from situation to situation are called *relational fluents*. They are denoted by predicate symbols taking a situation term as their last argument. Similarly, functions whose values vary from situation to situation are called *functional fluents*, and are denoted by function symbols taking a situation term as their last argument. For example, $isCarrying(robot, p, s)$, meaning that a *robot* is carrying package p in situation s, is a relational fluent; $location(robot, s)$, denoting the location of *robot* in situation s, is a functional fluent. For simplicity, we shall not treat functional fluents in this paper.

To axiomatize the primitive actions and fluents of a domain of application, one must provide the following axioms:

1. Action precondition axioms, one for each primitive action $A(\vec{x})$, having the syntactic form
$$Poss(A(\vec{x}), s) \equiv \Pi_A(\vec{x}, s),$$
where $\Pi_A(\vec{x}, s)$ is a formula with free variables among $\vec{x}, s$, and whose only situation term is s. Action precondition axioms characterize (via the formula $\Pi_A(\vec{x}, s)$) the conditions under which it is possible to execute action $A(\vec{x})$ in situation s. In addition to these, one must provide suitable unique names axioms for actions.
2. Successor state axioms, one for each fluent F, having the syntactic form
$$F(\vec{x}, do(a, s)) \equiv \Phi_F(\vec{x}, a, s),$$
where $\Phi_F(\vec{x}, a, s)$ is a formula with free variables among $\vec{x}, a, s$, and whose only situation term is s. Successor state axioms embody the solution to the frame problem of Reiter [21].
3. Axioms describing the initial situation – what is true initially, before any actions have occurred. This is any finite set of sentences that mention only the situation term S_0, or that are situation independent.

2.1 Golog

The "traditional" approach to high-level robotic control is to provide suitable goals, derive plans for achieving these goals, then have the robot execute these plans. Planning, however, is known to be computationally intractable in general, and in any case, is out of the question for deriving complex behaviors involving many hundreds, and possibly thousands of actions. The perspective being pursued by the Cognitive Robotics Group at the University of Toronto is to reduce the reliance on *planning* for eliciting inter-

[1] We allow nondeterministic programs, so that, even by itself, this idea of an on-line execution of a program is problematic. See Section 3 below.

esting robot behaviors, and instead provide the robot with *programs* written in a suitable high-level language [16], in our case, *Golog* or *ConGolog*. As presented in [17] and extended in [8], *Golog* is a logic-programming language whose primitive actions are those of a background domain theory. Typically *Golog* programs are intended to be executed *off-line*, and then the sequence of actions returned by this off-line computation is executed on-line. Here we consider a variant of *Golog* that is intended to be executed entirely *on-line* [9]. It includes the following constructs:

nil,	empty program
a,	primitive action
$\phi?$,	test the truth of condition ϕ
$(\delta_1 ; \delta_2)$,	sequence
$(\delta_1 \mid \delta_2)$,	nondeterministic choice of two actions
$\pi v.\delta$,	nondeterministic choice of argument to an action
δ^*,	nondeterministic iteration
proc $P(\vec{v})\,\beta$ **end**,	procedure with formal parameters $\vec{v}$ and body β.

In contrast to straight line or partially ordered plans a *Golog* program can be arbitrary complex, including loops, recursive procedures and nondeterministic choice.

Example 2.1 The following is a blocks world Golog program that nondeterministically builds a tower of blocks spelling "paris" or "rome". In turn, the procedure for building a Rome tower nondeterministically determines a block with the letter "e" that is clear and on the table, then nondeterministically selects a block with letter "m" and moves it onto the "e" block, etc. There is a similar procedure for *makeParis*; neither procedure has any parameters.

```
proc tower   makeParis | makeRome   endProc.
proc makeRome
  $\pi\, b_0.[e(b_0) \wedge ontable(b_0) \wedge clear(b_0)]?$ ;
    $\pi\, b_1.m(b_1)?\, ; move(b_1, b_0)$ ;
      $\pi\, b_2.o(b_2)?\, ; move(b_2, b_1)$ ;
        $\pi\, b_3.r(b_3)?\, ; move(b_3, b_2)$
endProc

proc makeParis
  $\pi\, b_0.[s(b_0) \wedge ontable(b_0) \wedge clear(b_0)]?$ ;
    $\pi\, b_1.i(b_1)?\, ; move(b_1, b_0)$ ;
      $\pi\, b_2.r(b_2)?\, ; move(b_2, b_1)$ ;
        $\pi\, b_3.a(b_3)?\, ; move(b_3, b_2)$
          $\pi\, b_4.p(b_4)?\, ; move(b_4, b_3)$
endProc
```

As in [8], we associate to programs a *transition semantics*, i.e. a semantics based on single steps of program execution. Informally, this semantics declares that as a program proceeds, a program counter moves from the very beginning of the program along its intermediate states. A *configuration* is a pair consisting of a program state (the part of the original program that is left to perform) and a situation.

To specify this semantics, we introduce two predicates *Trans* and *Final*.

- $Trans(\delta, s, \delta', s')$, given a program δ and a situation s, tells us which is a possible next step in the computation, returning the resulting situation s' and the program δ' that remains to be executed. In other words, $Trans(\delta, s, \delta', s')$ denotes a transition relation between configurations.
- $Final(\delta, s)$ tells us whether a configuration (δ, s) can be considered *final*, that is whether the computation is completed (no program remains to be executed). We have $Final(nil, s)$, but also $Final(\delta^*, s)$ since δ^* requires 0 or more repetitions of δ and so it is possible not to execute δ at all, completing the program immediately.

Trans

The predicate *Trans* is characterized by the following axioms:

1. Empty program:
$$Trans(nil, s, \delta', s') \equiv False$$
2. Primitive actions:
$$Trans(a, s, \delta', s') \equiv Poss(a, s) \wedge \delta' = nil \wedge s' = do(a, s)$$
3. Test actions:[2]
$$Trans(\phi?, s, \delta', s') \equiv \phi[s] \wedge \delta' = nil \wedge s' = s$$
4. Sequence:
$$Trans(\delta_1;\delta_2, s, \delta', s') \equiv \exists\gamma. Trans(\delta_1, s, \gamma, s') \wedge \delta' = \gamma;\delta_2 \vee Final(\delta_1, s) \wedge Trans(\delta_2, s, \delta', s')$$
5. Nondeterministic choice:
$$Trans(\delta_1 \mid \delta_2, s, \delta', s') \equiv Trans(\delta_1, s, \delta', s') \vee Trans(\delta_2, s, \delta', s')$$
6. Pick:
$$Trans(\pi v.\delta, s, \delta', s') \equiv \exists x. Trans(\delta^v_x, s, \delta', s')$$ [3]
7. Iteration:
$$Trans(\delta^*, s, \delta', s') \equiv \exists\gamma. Trans(\delta, s, \gamma, s') \wedge \delta' = \gamma; \delta^*$$

[2]We write ϕ to denote a term representing a situation calculus formula with suppressed situational argument and $\phi[s]$ to denote the formula with the restored argument. We assume any standard way of encoding first-order situation calculus formulas.

[3]Here, δ^v_x is the program resulting from substituting x for v uniformly in δ.

The assertions above characterize when a configuration (δ, s) can evolve (in a single step) to a configuration (δ', s'). Intuitively they can be read as follows:

1. (nil, s) cannot evolve to any configuration.
2. (a, s) evolves to $(nil, do(a, s))$, provided it is possible to execute a in s. Notice that after having performed a, nothing remains to be performed.
3. $(\phi?, s)$ evolves to (nil, s), provided that $\phi[s]$ holds. Otherwise, it cannot proceed. Notice that in any case the situation remains unchanged.
4. $(\delta_1;\delta_2, s)$ can evolve to $(\delta_1';\delta_2, s')$, provided that (δ_1, s) can evolve to (δ_1', s'). Otherwise, it can evolve to (δ_2', s'), provided that (δ_1, s) is a final configuration and (δ_2, s) can evolve to (δ_2', s').
5. $(\delta_1|\delta_2, s)$ can evolve to (δ', s'), provided that either (δ_1, s) or (δ_2, s) can do so.
6. $(\pi v.\delta, s)$ can evolve to (δ', s'), provided that there exists an x such that (δ_x^v, s) can evolve to (δ', s').
7. (δ^*, s) can evolve to $(\delta';\delta^*, s')$ provided that (δ, s) can evolve to (δ', s'). Observe that (δ^*, s) can also not evolve at all, because (δ^*, s) is final by definition (see below).

To simplify the discussion, we have omitted axioms for procedures. These can be found in the extended version of [8].

Final

The predicate *Final* is characterized by the following axioms:

1. Empty program:
$$Final(nil, s) \equiv True$$
2. Primitive action:
$$Final(a, s) \equiv False$$
3. Test action:
$$Final(\phi?, s) \equiv False$$
4. Sequence:
$$Final(\delta_1;\delta_2, s) \equiv Final(\delta_1, s) \wedge Final(\delta_2, s)$$
5. Nondeterministic choice:
$$Final(\delta_1 \mid \delta_2, s) \equiv Final(\delta_1, s) \vee Final(\delta_2, s)$$
6. Pick:
$$Final(\pi v.\delta, s) \equiv \exists x.Final(\delta_x^v, s)$$
7. Iteration:
$$Final(\delta^*, s) \equiv True$$

*Trans** **and** *Do*

The possible configurations that can be reached by a program δ starting in a situation s are those obtained by following repeatedly the transition relation denoted by *Trans* starting from (δ, s), i.e. those in the reflexive transitive closure of the transition relation. Such a relation, denoted by *Trans**, is defined as the (second-order) situation calculus formula:

$$Trans^*(\delta, s, \delta', s') \equiv \forall T[\ldots \supset T(\delta, s, \delta', s')]$$

where ... stands for the conjunction of the universal closure of the following two sentences:

$$T(\delta, s, \delta, s)$$
$$Trans(\delta, s, \delta'', s'') \wedge T(\delta'', s'', \delta', s') \supset T(\delta, s, \delta', s')$$

Using *Trans** and *Final* we can give a new definition of the *Do* relation of [17] as:

$$Do(\delta, s, s') \equiv \exists\delta'.Trans^*(\delta, s, \delta', s') \wedge Final(\delta', s').$$

In other words, $Do(\delta, s, s')$ holds iff it is possible to repeatedly single-step the program δ, obtaining a program δ' and a situation s' such that δ' can legally terminate in s'.

3 On vs. Off-Line Golog Interpreters

Before describing our approach to execution monitoring, we must first distinguish carefully between on-line and off-line *Golog* interpreters.[4] The relation $Do(\gamma, s, s')$ means that s' is a terminating situation resulting from an execution of program γ beginning with situation s. This relation has a natural Prolog implementation in terms of the one-step interpreter *trans*:

```
offline(Prog,S0,Sf) :- final(Prog,S0), S0 = Sf ;
                       trans(Prog,S0,Prog1,S1),
                       offline(Prog1,S1,Sf).
```

A Brave On-Line Interpreter

The difference between on- and off-line interpretation of a *Golog* program is that the former must select a first action from its program, commit to it (or, in the physical world, do it), then repeat with the rest of the program. The following is such an interpreter:

```
online(Prog,S0,Sf) :- final(Prog,S0), S0 = Sf ;
  trans(Prog,S0,Prog1,S1),      /* Select a first
                                  action of Prog. */
  !,                /* Commit to this action. */
  online(Prog1,S1,Sf).
```

[4]An on-line interpreter based on *Trans* and *Final* was originally proposed in [9] to give an account of Golog /ConGolog programs with sensing actions. Here we make use of a simplified on-line interpreter that does not deal with sensing actions, but is suitable for coupling with an execution monitor.

The on and off-line interpreters differ only in the latter's use of the Prolog cut (!) to prevent backtracking to `trans` to select an alternative first action of `Prog`.[5] The effect is to commit to the first action selected by `trans`. We need this because a robot cannot undo any actions that it has actually performed in the physical world. It is this commitment that qualifies the clause to be understood as on-line interpreter. We refer to it as *brave* because it may well reach a dead-end, even if the program it is interpreting has a terminating situation.

A Cautious On-Line Interpreter

To avoid the possibility of following dead-end paths, one can define a *cautious* on-line interpreter as follows:

```
online(Prog,S0,Sf) :- final(Prog,S0), S0 = Sf ;
  trans(Prog,S0,Prog1,S1),      /* Select a first
                                 action of Prog. */
  offline(Prog1,S1,S2),      /* Make sure the rest
                               of Prog terminates. */
  !,              /* Commit to this action. */
  online(Prog1,S1,Sf).
```

This is much more cautious than its brave counterpart; it commits to a first action only if that action is guaranteed to lead to a successful off-line termination of the program. Provided this program has a terminating situation, a cautious on-line interpreter never reaches a dead-end.

A cautious on-line interpreter appeals to the off-line execution of the robot's program (in the process of guaranteeing that after committing to a program action, the remainder of the program terminates). Therefore, this requirement precludes cautious interpretation of robot programs that appeal to sensing actions [15], since such actions cannot be performed off-line.[6] Because the brave interpreter never looks ahead, it is suitable for programs with sense actions. The price it pays for this is a greater risk of following dead-end paths.

Committing to an action is an intrinsically *procedural* notion, and so it is highly desirable, in any logical approach to modeling dynamical systems, to very tightly delimit where in the theory and implementation this nonlogical notion appears. In our case, we can point to the Prolog cut operator in the above on-line interpreters as the exact point at which the procedural notion of commitment is realized.

The above interpreters are implemented in Prolog, and are lifted directly from *Final*, *Trans*, and *Do* introduced above. Such interpreters require that the domain specific action precondition and successor state axioms, and axioms about the initial situation, be expressible as Prolog clauses. Therefore, our implementation inherits Prolog's Closed World Assumption, but this is a limitation of the implementation, not the general theory. The full version of the cautious on-line interpreter can be found in [10].

[5]Keep in mind that *Golog* programs may be nondeterministic.

[6]However, one could imagine a cautious interpreter that verifies off-line that the program terminates for all possible outcomes of its sensing actions. Even better, perhaps the programmer has already proved this.

4 Execution Monitoring of Golog Programs

In this section we give a situation calculus specification for the behavior of a Golog program under the influence of an execution monitor. We first provide a very general framework, without committing to any particular details of the monitor. Then we describe one specific monitor that forms the basis for the implementation of Section 5 below.

4.1 The General Framework

Here we discuss how on-line interpretation of Golog programs can be combined with a monitor. We imagine that after executing a primitive action or evaluating a test condition, a robot compares its mental world model with reality. We assume that all discrepancies between the robot's mental world and reality are the result of exogenous actions, and moreover, that the robot observes all such actions.[7] It will be the execution monitor that observes whether an exogenous action has changed the values of one or several fluents and, if necessary, recovers from this unanticipated event. This cycle of on-line interpreting, sensing and recovering (if necessary) repeats until the program terminates.

Just as we specified a semantics, via *Trans*, for *Golog* programs in Section 2.1, we want now to specify such

[7]A similar idealization about the observability of all exogenous events is a common assumption in discrete event control theory (e.g. [20, 6]). On the face of it, this idealization seems dubious in practice. One can argue convincingly that agents never observe action *occurrences* – Fido ate the sandwich – only their *effects* – The sandwich is no longer on the table. One can reconcile both points of view by supposing that instead of directly sensing exogenous actions, the robot can sense only the truth values of certain fluents. One can then introduce a set of new fictitious actions, one for each such fluent, whose effects are to alter their corresponding fluents' truth values. The robot can compute, from its successor state axioms, what fluents hold in its mental world. Now, when the robot observes the values of some its fluents in the physical world, it compares them with their values in its mental world; all discrepancies, if any, can be determined directly. Then, it can determine which fictitious actions must have "occurred" to account for the observed discrepancies between the physical world and the robot's mental world. It is these (inferred) fictitious actions that assume the role of the observable exogenous actions mentioned above.

a semantics for *Golog* programs *with* execution monitoring. Our definition will parallel that of Section 2.1. This closed-loop system (online interpreter and execution monitor) is characterized formally by a new predicate symbol $TransEM(\delta, s, \delta', s')$, describing a one-step transition consisting of a single $Trans$ step of program interpretation, followed by a process, called $Monitor$, of execution monitoring. The role of the execution monitor is to get new sensory input in the form of an exogenous action and (if necessary) to generate a program to counter-balance any perceived discrepancy. As a result of all this, the system passes from configuration (δ, s) to configuration (δ', s') specified as follows:

$$TransEM(\delta, s, \delta', s') \equiv \exists \delta'', s''.Trans(\delta, s, \delta'', s'') \wedge Monitor(\delta'', s'', \delta', s').$$

This is a brave version of $TransEM$. Its cautious counterpart is:

$$TransEM(\delta, s, \delta', s') \equiv \exists \delta'', s''.Trans(\delta, s, \delta'', s'') \wedge \exists s'''.Do(\delta'', s'', s''') \wedge Monitor(\delta'', s'', \delta', s').$$

The possible configurations that can be reached by a program δ from a situation s with execution monitoring are those obtained by repeatedly following $TransEM$ transitions, i.e. those in the reflexive transitive closure of this relation.

As we did for implementing on-line *Golog* interpreters (Section 3), we can now describe brave and cautious versions of on-line *Golog* interpreters *with* execution monitoring.

Brave On-Line Execution Monitor

```
onlineEM(Prog,S0,Sf) :- final(Prog,S0), S0 = Sf ;
  trans(Prog,S0,Prog1,S1),
  !,
  monitor(Prog1,S1,Prog2,S2), !,
  onlineEM(Prog2,S2,Sf).
```

Cautious On-Line Execution Monitor

```
onlineEM(Prog,S0,Sf) :- final(Prog,S0), S0 = Sf ;
  trans(Prog,S0,Prog1,S1),
  offline(Prog1,S1,S),
  !,
  monitor(Prog1,S1,Prog2,S2), !,
  onlineEM(Prog2,S2,Sf).
```

Next, we focus on the monitor. Let *exo* be an exogenous event, which might be as simple as a primitive action, or as complex as an arbitrary *Golog* program. We specify the behavior of our generic monitor by:

$$\begin{aligned} Monitor(\delta, s, \delta', s') \equiv \exists exo.Do(exo, s, s') \wedge \\ [\neg Relevant(\delta, s, s') \wedge \delta' = \delta \vee \\ Relevant(\delta, s, s') \wedge Recover(\delta, s, s', \delta')]. \end{aligned} \tag{1}$$

Here, $Relevant(\delta, s, s')$ is a predicate that specifies whether the discrepancy between s and s' is relevant in the current state δ of the program. If this discrepancy does not matter – $\neg Relevant(\delta, s, s')$ – then the execution monitor takes no action – $\delta' = \delta$. Otherwise, the monitor should recover from the exogenous action. The predicate $Recover(\delta, s, s', \delta')$ provides for this by determining a new program, δ', whose execution in situation s' is intended to achieve an outcome equivalent (in a sense left open for the moment) to that of program δ, had the exogenous event not occurred.

A wide range of monitors can be achieved by defining *Relevant* and *Recover* in different ways. In the next section we elaborate on one such choice, one that will form the basis of the implementation of Section 5.

4.2 A Specific Monitor

Now we develop a simple realization of the above general framework, by fixing on particular predicates *Relevant* and *Recover*.

We begin by assuming that for each application domain a programmer provides:

1. The specification of all primitive actions (robot's and exogenous) and their effects, together with an axiomatization of the initial situation, as described in Section 2.
2. A *Golog* program that may or may not take into account exogenous actions occurring when the robot executes the program. We shall assume that this program has a particular form, one that takes into account the programmer's *goal* in writing it. Specifically, we assume that along with her program, the programmer provides a first order sentence describing the program's goal, or what programmers call a program *postcondition*. We assume further that this postcondition is postfixed to the program. In other words, if δ is the original program, and *goal* is its postcondition, then the program we shall be dealing with in this paper will be $\delta ; goal?$. This may seem a useless thing to do whenever δ is known to satisfy its postconditions, but as we shall see below, our approach to execution monitoring will change δ, and we shall need a guarantee that whenever the modified program terminates, it does so in a situation satisfying the original postcondition.

Next, we take $Relevant(\delta, s, s')$ to be $\neg\exists s'' Do(\delta, s', s'')$, so that the definition (1) of $Monitor$ becomes:

$$\begin{aligned} Monitor(\delta, s, \delta', s') \equiv \exists exo.Do(exo, s, s') \wedge \\ [\exists s'' Do(\delta, s', s'') \wedge \delta' = \delta \vee \\ \neg\exists s'' Do(\delta, s', s'') \wedge Recover(\delta, s, s', \delta')]. \end{aligned}$$

$Monitor$ checks for the existence of an exogenous program, determines the situation s' reached by this program, and if the monitored program δ terminates off-line, the monitor returns δ, else it invokes a recovery mechanism to determine a new program δ'. Therefore, $Monitor$ appeals to $Recover$ only as a last resort; it prefers to let the monitored program keep control, so long as this is guaranteed to terminate off-line in a situation where the program's goal holds. (Remember that this goal has been postfixed to the original program, as described in 2 above.)

It only remains to specify the predicate $Recover(\delta, s, s', \delta')$ that is true whenever δ is the current state of the program being monitored, s is the situation prior to the occurrence of the exogenous program, s' is the situation after the exogenous event, and δ' is a new program to be executed on-line in place of δ, beginning in situation s'. We adopt the following specifications of *Recover*; it forms the basis of the implementation to be described later:

$$\begin{aligned}&Recover(\delta, s, s', \delta') \equiv\\ &\quad \exists p.straightLineProg(p) \wedge\\ &\qquad \exists s''.Do(p\,;\delta, s', s'') \wedge \delta' = p\,;\delta \wedge\\ &\qquad [\forall p', s''.straightLineProg(p') \wedge Do(p'\,;\delta, s', s'') \supset\\ &\qquad\qquad length(p) \leq length(p')].\end{aligned}$$

Here, the recovery mechanism is conceptually quite simple; it determines a shortest straight-line program p such that, when prefixed onto the program δ, yields a program that terminates off-line. This is quite easy to implement; in its simplest form, simply generate all length one prefixes, test whether they yield a terminating off-line computation, then all length two prefixes, etc, until one succeeds, or some complexity bound is exceeded.[8] Notice that here we are appealing to the assumption 2 above that all monitored programs are postfixed with their goal conditions. We need something like this because the recovery mechanism *changes* the program being monitored, by adding a prefix to it. The resulting program may well terminate, but in doing so, it may behave in ways unintended by the programmer. But so long as the goal condition has been postfixed to the original program, all terminating executions of the altered program will still satisfy the programmer's intentions.

One disadvantage of the above recovery mechanism is that it will not recognize instances of exogenous events that happen to *help* in achieving the goal condition. In the extreme case of this, an exogenous event might create a situation that actually satisfies the goal. The above recovery procedure, being blind to such possibilities, will unthinkingly modify the current program state by prefixing to it a suitable plan, and execute the result, despite the fact that in reality, it is already where it wants to be. In effect, the recovery procedure has a built-in assumption that all exogenous events, if not neutral with respect to achieving the goal, are malicious.

[8]One can imagine much more sophisticated realizations of this simple idea that make use of the actions performed by *exo*, but we do not pursue this topic here.

5 An Implementation

The above theory of execution monitoring is supported by an implementation, in Prolog, that we demonstrate here for the blocks world program of Example 2.1. We use the cautious on-line monitor of Section 4.1, and a straightforward implementation of *Monitor* and *straightLineProg*(p). The Prolog code is provided in [10].

5.1 A Blocks World Example

In this section, the blocks world is axiomatized with successor state and action precondition axioms. We use the following function and predicate constants.

Actions

- $move(x, y)$: Move block x onto block y, provided both are clear.
- $moveToTable(x)$: Move block x onto the table, provided x is clear and is not on the table.

Fluents

- $On(x, y, s)$: Block x is on block y, in situation s.
- $Clear(x, s)$: Block x has no other blocks on top of it, in situation s.
- $Ontable(x, s)$: Block x is on the table in s.

Other predicate constants

The predicates $R(b)$, $O(b)$, $M(b)$, $E(b)$, $P(b)$, $A(b)$, $I(b)$, $S(b)$ are true when their arguments are blocks with the corresponding letters on them.

Successor state axioms

$$\begin{aligned}&On(x, y, do(a, s)) \equiv a = move(x, y) \vee On(x, y, s) \wedge\\ &\quad a \neq moveToTable(x) \wedge \neg(\exists z) a = move(x, z).\end{aligned}$$

$$\begin{aligned}&Ontable(x, do(a, s)) \equiv a = moveToTable(x) \vee\\ &\quad Ontable(x, s) \wedge \neg(\exists y) a = move(x, y).\end{aligned}$$

$$\begin{aligned}&Clear(x, do(a, s)) \equiv (\exists y, z).[\,a = move(y, z) \vee\\ &\quad a = moveToTable(y))\,] \wedge On(y, x, s) \vee\\ &\qquad Clear(x, s) \wedge \neg(\exists w) a = move(w, x).\end{aligned}$$

Action precondition axioms

$$\begin{aligned}&Poss(move(x, y), s) \equiv Clear(x, s) \wedge\\ &\quad Clear(y, s) \wedge x \neq y.\end{aligned}$$

$$\begin{aligned}&Poss(moveToTable(x), s) \equiv Clear(x, s) \wedge\\ &\quad \neg Ontable(x, s).\end{aligned}$$

Unique names axioms for actions

$$move(x, y) \neq moveToTable(x)$$

$$move(x, y) = move(x', y') \supset x = x' \wedge y = y'$$

$$moveToTable(x) = moveToTable(x') \supset x = x'$$

In our example, the initial situation is such that all blocks are on the table and clear (see Figure 1). There is no block with the letter "p"[9], but there are several blocks with letters for spelling "aris" and "rome", as well as blocks with letters "n" and "f" (which are irrelevant to building a tower spelling "rome" or "paris").

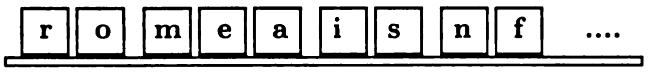


Figure 1: Part of the initial situation.

The program goal is:

$$Goal(s) \equiv SpellsParis(s) \vee SpellsRome(s),$$

$$SpellsRome(s) \equiv (\exists b_0, b_1, b_2, b_3).\ R(b_3) \wedge O(b_2) \wedge M(b_1) \wedge E(b_0) \wedge Ontable(b_0, s) \wedge On(b_1, b_0, s) \wedge On(b_2, b_1, s) \wedge On(b_3, b_2, s) \wedge Clear(b_3, s).$$

$$SpellsParis(s) \equiv (\exists b_0, b_1, b_2, b_3, b_4).\ P(b_4) \wedge A(b_3) \wedge R(b_2) \wedge I(b_1) \wedge S(b_0) \wedge Ontable(b_0, s) \wedge On(b_1, b_0, s) \wedge On(b_2, b_1, s) \wedge On(b_3, b_2, s) \wedge On(b_4, b_3, s) \wedge clear(b_4, s).$$

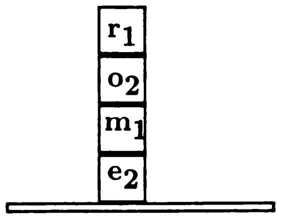


Figure 2: A goal arrangement of blocks.

Figure 2 represents an arrangement of blocks that satisfies the program goal.

An implementation, in Eclipse Prolog, is provided in [10].

5.2 An Execution Trace

The original procedure *tower* is very simple and was not designed to respond to external disturbances of any kind. However, as the trace demonstrates, the execution monitor is able to produce fairly sophisticated behavior in response to unforeseen exogenous events.

In Golog, tests do not change the situation, but all other primitive actions do. Each time the program performs a primitive action or evaluates a test, an exogenous program may occur. In the example below, any sequence of actions that is possible in the current situation can be performed by an external agent. This is realized in the implementation by interactively asking the user to provide exogenous events after each elementary program operation (test or primitive action).

The following is an annotated trace of the first several steps of our implementation for this blocks world setting. We use `this font` for the actual output of the program and *italics* for states of the *Golog* program *tower* of Example 2.1. The symbol ":" in the Prolog implementation of the interpreter corresponds to sequential composition ";" in Golog and Prolog's term "pi" corresponds to π in Golog, etc.

```
[eclipse] onlineEM((tower : ?(goal)), s0, S).

 Program state = (nil: pi(b1,?(m(b1)) : move(b1,e1):
    pi(b2,?(o(b2)) : move(b2,b1) : pi(b3,?(r(b3)) :
        move(b3,b2))))) : ?(goal)
 Current situation: s0
```

The cautious interpreter first tried to execute *makeParis* off-line. This failed because there is no "p" block. It then proceeded with *makeRome*.[10] According to the implementation of a cautious on-line interpreter (see section 3), *Trans* does the first step of *makeRome*:

$\pi\, b_0.[e(b_0) \wedge ontable(b_0) \wedge clear(b_0)]?\,;$

by determining that the block e_1 will be the base of a goal tower. The remainder of the program (the "program state" in the output above) is the following:

nil ;
$\pi\, b_1.m(b_1)?\,; move(b_1, e1)\,;$
  $\pi\, b_2.o(b_2)?\,; move(b_2, b_1)\,;$
   $\pi\, b_3.r(b_3)?\,; move(b_3, b_2)$; $(goal)?$

where *nil* results after the first step (see axioms of *Trans* for $\pi v.\delta$ and $\phi?$).

```
>Enter: an exogenous program or noOp if none occurs.
        move(n,m1) : move(f,n) : move(i2,o3).

 No recovery necessary. Proceeding with the next
 step of program.

  Program state = (nil : move(m2, e1) :
     pi(b2, ?(o(b2)) : move(b2, m2) :
         pi(b3, ?(r(b3)) : move(b3, b2)))) : ?(goal)
  Current situation: do(move(i2, o3), do(move(f, n),
                     do(move(n, m1), s0)))
```

The first exogenous program covered blocks m_1, n and o_3 (see Fig. 3), but the remaining program

nil ;
$move(m_2, e_1)\,;$ $\surd$

[9]We selected this arrangement intentionally to illustrate the difference between cautious and brave interpreters.

[10]A brave interpreter would have eventually failed, without even trying *makeRome*.

$\pi\, b_2.o(b_2)?\,; move(b_2, m_2)\,;$
$\quad \pi\, b_3.r(b_3)?\,; move(b_3, b_2)\ ;\ ?(goal)$

can still be successfully completed because there remain enough uncovered blocks of the right kind to construct "rome", so it continues.

```
>Enter: an exogenous program or noOp if none occurs.
        move(i1,o1) : move(r2,o2).
Start recovering...

 New program = moveToTable(r2) :
     (nil : move(m2,e1) : pi(b2,?(o(b2)) :
         move(b2,m2) : pi(b3,?(r(b3)) :
             move(b3,b2)))) : ?(goal)

 Current situation: do(move(r2,o2),do(move(i1,o1),
                 do(move(i2,o3), do(move(f,n),
                    do(move(n,m1),s0)))))
```

After the second exogenous program move(i1,o1) : move(r2,o2), all three blocks with letter "o" are covered (see Figure 4).

Because it is not possible to move blocks o_1, o_2, o_3 (by the precondition axiom for $move(x, y)$), the remaining program cannot be completed. Hence, the *Monitor* gives control to *Recover* that, with the help of the planner *straightLineProg*(p), determines a shortest corrective sequence p of actions (namely moveToTable(r2)) in order to allow the program to resume, and prefixes this action to the previous program state:

$moveToTable(r2)\,; nil\ ;$
$\quad move(m_2, e_1)\,;$
$\quad\quad \pi\, b_2.o(b_2)?\,; move(b_2, m_2)\,;$
$\quad\quad\quad \pi\, b_3.r(b_3)?\,; move(b_3, b_2)\ ;\ ?(goal)$

From this point on, the on-line evaluation continues by doing one step of the new program. If after that, no exogenous disturbances occur during the next two steps of the execution of this new program, it will reach the following program state:

$$\begin{array}{l} nil\,; \\ \quad move(o2, m2)\,; \\ \quad\quad \pi\, b_3.r(b_3)?\,; move(b_3, o_2)\,;\ ?(goal) \end{array} \qquad (2)$$

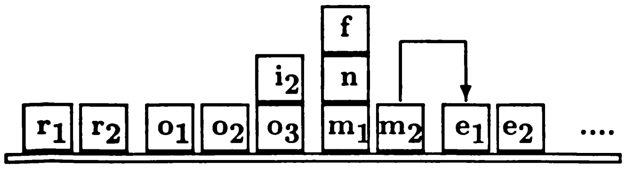


Figure 3: The first exogenous disturbance occurred when the program was ready to move m_2 on top of e_1.

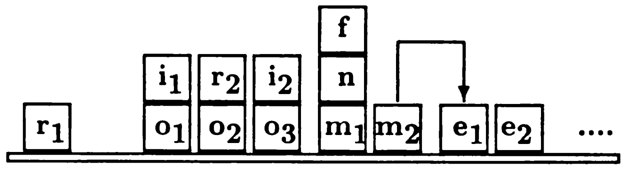


Figure 4: The second exogenous disturbance.

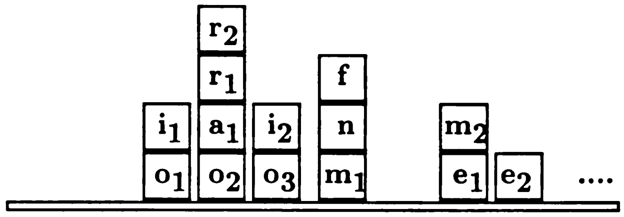


Figure 5: The pile of blocks covers the block o_2.

Let assume that at this point a third exogenous program occurs (see Figure 5):

```
>Enter: an exogenous program or noOp if none occurs.
        move(a1,o2) : move(r1,a1) : move(r2,r1).
Start recovering...
  New program = (moveToTable(r2) :
      moveToTable(r1) : moveToTable(a1)) :
         (nil : move(o2, m2) : pi(b3, ?(r(b3)) :
            move(b3, o2))) : ?(goal)

  Program state = (nil : moveToTable(r1) :
     moveToTable(a1)) : (nil : move(o2, m2) :
        pi(b3, ?(r(b3)) : move(b3, o2))) : ?(goal)

  Current situation: do(moveToTable(r2),
    do(move(r2, r1), do(move(r1, a1),
      do(move(a1, o2), do(move(m2, e1),
        do(moveToTable(r2), do(move(r2, o2),
         do(move(i1, o1), do(move(i2, o3),
           do(move(f,n), do(move(n,m1), s0)))))))))))
```

The recovery procedure determined that the sequence of three corrective actions moveToTable(r2), moveToTable(r1), moveToTable(a1) will lead to a situation where the program (2) can be resumed, and moreover, this is a shortest such correction. If no other exogenous actions happen, the program will eventually successfully terminate, having built a tower for "rome".

6 Correctness Properties for Execution Monitors

With definitions for *TransEM* and *Monitor*, as in Section 4.1, it becomes possible to formulate, and ultimately prove, various correctness properties for execution monitors. These properties are intended to capture suitable concepts of ***controllability*** following the intuition behind similar concepts introduced for discrete event systems [20]. Informally, controllability is the property that characterizes a closed-loop system (a *Golog* program coupled with the execution monitor): this is the ability of a monitored program to behave correctly even if exogenous actions occur during the robot's execution of the program. There are many possible definitions, with varying degrees of generality, of what counts as correct behavior of a monitored system. We focus here on various correctness properties one might want to prove of the monitor.

Recall that the general execution monitor, as specified by (1), is the relation $Monitor(\delta, s, s', \delta')$, meaning that whenever δ is the state of monitored program in situation s, and s' is a situation resulting from an exogenous event occurrence at s, then the on-line execution should resume in s' with the new program δ'. Then, by analogy with Floyd-Hoare style correctness and termination properties, we can formulate a variety of verification tasks, some examples of which we now describe. These are parameterized by two predicates:

1. $P(s)$, a desirable property that a terminating situation s must satisfy. For example, P might describe a postcondition of the program being monitored.
2. $Q(\delta, s, s')$, a relationship between the current program state δ and s, the current situation, and s', the situation resulting from an exogenous event occurring in s. For example, Q might express that δ terminates off-line when executed in s and also when executed in s'.

Weak Termination and Correctness

$$(\forall \delta, s, s').Q(\delta, s, s') \supset Monitor(\delta, s, s', \delta') \supset (\exists s'').Do(\delta', s', s'') \wedge P(s'').$$

The task here is to verify that, under condition Q, whenever *Monitor* determines a new program with which to resume the system computation after an exogenous event occurrence, that program has a terminating (off-line) computation resulting in a final situation in which P holds.

Strong Termination and Correctness

$$(\forall \delta, s, s').Q(\delta, s, s') \supset (\exists \delta').Monitor(\delta, s, s', \delta') \wedge (\exists s'').Do(\delta', s', s'') \wedge P(s''),$$

Under condition Q, *Monitor* always determines a new program with which to resume the system computation after an exogenous event occurrence, and that program has a terminating (off-line) computation resulting in a final situation in which P holds.

Even Stronger Termination and Correctness

$$(\forall \delta, s, s').Q(\delta, s, s') \supset (\exists \delta').Monitor(\delta, s, s', \delta') \wedge (\exists s'').Do(\delta', s', s'') \wedge (\forall s'').Do(\delta', s', s'') \supset P(s'').$$

Here, the correctness property is that under condition Q, *Monitor* always determines a new program that terminates off-line, and all these terminating situations satisfy P.

It is also possible to formulate various correctness properties for the entire monitored system, for example, the weak property that provided the monitored program terminates, then it does so in a desirable situation:

$$(\forall \delta, s, s').TransEM^*(\delta, s, nil, s') \supset P(s'),$$

where $TransEM^*$ is the transitive closure of $TransEM$.

Other variations on the above themes are possible, but our purpose here is not to pursue these issues in depth, but simply to point out that correctness properties for monitored systems are easily formulated within our framework. Moreover, because this framework is entirely within the situation calculus, such correctness proofs can be constructed totally within a classical logic.

7 Discussion

There are several systems designed to interleave monitoring with plan execution: PLANEX [7], IPEM [1], ROGUE [13], SPEEDY [3]. We differ from these and similar proposals, first by the formal neatness of our approach, secondly by the fact that ours is a story for monitoring arbitrary *programs*, not simply straight line or partially ordered plans. Moreover, we do not assume that the monitored plan is generated automatically from scratch, but rather that it has been provided by a programmer.

In a sequence of papers [25, 26, 27] Schoppers proposes and defends the idea of "universal plans", which "address the tension between reasoned behavior and timely response by caching reactions for classes of possible situations". From our point of view, the notion of a universal plan is closely related to the notion of controllable languages developed for discrete event systems control [20]. There, a language (a set of linear plans) is controllable iff the effects of all possible uncontrollable events do not lead outside the set of plans that this language contains. In other words, just as for Schoppers, all required system reactions to possible contingencies are compiled into the controllable language. Our framework is different, but complementary; it favors the on-line generation of appropriate reactions to exogenous events, as opposed to precompiling them into the *Golog* program. *ConGolog* [8, 9] is a much richer version of *Golog* that supports concurrency, prioritized interrupts and exogenous actions. Reactive behaviors are easily representable by ConGolog's interrupt mechanism, so that a combination of reactive behaviors with "deliberative" execution monitoring is possible. This would allow one to experiment with different mixtures of execution monitoring and reactivity, with the advantage of preserving the unifying formal framework of the situation calculus, but this remains an open research problem.

The theory of embedded planning [29, 12, 30] introduces notions of planning with failure and has motivations very similar to ours. The authors propose several formal languages that, like *Golog*, include constructs for sequence, conditionals, loops and recursion. The emphasis is on reactive programs, but their proposal does provide for replanning during execution.

Several authors rely on formal theories of actions for the purposes of characterizing appropriate notions of action failures [2, 24], but they do not consider execution monitoring per se.

Perhaps the most sophisticated existing plan execution monitor is the XFRM system of Beetz and McDermott [4, 5]. This provides for the continual modification of robot plans (programs) during their execution, using a rich collection of failure models and plan repair strategies. Nothing in our proposal so far can rival the functionality of the XFRM system. The primary objective of our approach is to provide compact, declarative representations for the entire process of execution, monitoring and recovery from failure, and this paper has presented our framework for this research program. It remains to see whether the logical purity of our approach will pay off with clear, analyzable specifications that lead to implementations rivaling that of XFRM.

8 Conclusions and Future Work

We have presented a very general, situation calculus-based account of execution monitoring for high-level robot programs, and illustrated the theory with an implementation (in simulation mode) of a specific monitor. The approach has the advantage of being completely formal, and therefore is suitable for formulating, and ultimately proving, correctness properties for monitored systems.

Plans for ongoing and future work include the following issues:

1. Draw closer parallels with the concept of controllable systems in discrete event control theory [20, 6, 14].
2. Explore realizations of execution monitors different than that presented in Section 4.2 [28].
3. Investigate techniques for proving correctness properties of various monitors and monitored systems.
4. Investigate the concept of execution monitoring for non-terminating *Golog* programs [11].
5. Extend these ideas to temporal domains, for example, monitoring robot control programs written in sequential, temporal *Golog* [23].
6. Implement these ideas on the Cognitive Robotics Group's RWI B21 autonomous robot at the University of Toronto.

Acknowledgements:

This research was supported by grants from the Natural Sciences and Engineering Research Council of Canada, the Institute for Robotics and Intelligent Systems of the Government of Canada, and the Information Technology Research Centre of the Government of Ontario. Our thanks to the many people who contributed their ideas and suggestions during the evolution of our thinking about these issues: Yves Lespérance, Hector Levesque, Jeff Lloyd, John Mylopoulos, Brian Nixon, Fiora Pirri, Eric Sandewall, Murray Shanahan, Eugenia Ternovskaia, Murray Wonham.

References

[1] J. Ambrose-Ingerson and S. Steel. Integrating planning, execution and monitoring. In *AAAI-88*, p. 735–740. Morgan Kaufmann Publishers, San Francisco, CA, 1988.

[2] C. Baral, T. and Son. Relating theories of actions and reactive control. In *Robots, Softbots, Immobots: Theories of Action, Planning and Control*, working notes of the workshop held on July 27, 1997 in conjunction with the AAAI-97, Providence, Rhode Island.

[3] C. Bastié and P. Régnier. SPEEDY : Monitoring the Execution in Dynamic Environments. In Reasoning about Actions and Planning in Complex Environments, Proceedings of the Workshop at International Conference on Formal and Applied Practical Reasoning, Bonn, Germany, on June 4, 1996. Available as: Technical Report AIDA-96-11, Fachgebiet Intellektik, Technische Hochschule Darmstadt, Germany.

[4] M. Beetz and D. McDermott. Improving robot plans during their execution. In *2nd Int. Conf. on AI Planning Systems (AIPS-94)*, Kris Hammond (editor), p. 3–12, 1994.

[5] M. Beetz and D. McDermott. Expressing transformations of structured reactive plans. In *4th European Conference on Planning (ECP'97)*, Sam Steel (editor), p. 66–78, Toulouse, France, September 24-26, 1997.

[6] S.-L. Chung, S.Lafortune, F. Ling. Limited lookahead policies in supervisory control of discrete event systems. *IEEE Transactions on Automatic Control*, volume 37, N 12, p. 1921–1935, 1992.

[7] R.E. Fikes, P.E. Hart, and N.J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, volume 3, N 4, p. 251–288, 1972.

[8] G. De Giacomo, Y. Lespérance, and H.J. Levesque. Reasoning about concurrent executions, prioritized interrupts, and exogenous actions in the situation calculus. In *15th International Joint Conference on Artificial Intelligence (IJCAI-97)*, p. 1221–1226 1997.

[9] G. De Giacomo and H.J. Levesque. Congolog incremental interpreter. Technical report, Computer Science Department, University of Toronto, 1998. In preparation.

[10] G. De Giacomo, R. Reiter, M. Soutchanski Execution Monitoring of High-Level Robot Programs. In working notes of the 4th Symposium on *Logical Formalizations of Commonsense Reasoning*, London, UK, Jan. 6 - Jan. 9, 1998. Electronic version is available at: `www.ida.liu.se/ext/etai/nj/fcs-98/listing.html`

[11] G. De Giacomo, R. Reiter, E. Ternovskaia. Nonterminating processes in the situation calculus. In *Robots, Softbots, Immobots: Theories of Action, Planning and Control*, working notes of the workshop held on July 27, 1997 in conjunction with the AAAI-97, Providence, Rhode Island.

[12] F. Giunchiglia, P. Traverso, L. Spalazzi. Planning with failure. In 2nd International Conference on AI Planning Systems (AIPS-94), Chicago, IL, June 15-17, 1994.

[13] K.Z. Haigh and M. Veloso. Interleaving planning and robot execution for asynchronous user requests. In *Int. Conf. on Intelligent Robots and Systems (IROS-96)*, p. 148–155, November 1996, Osaka, Japan.

[14] J. Kosecka and R. Bajcsy. Discrete Event Systems for Autonomous Mobile Agents. *Robotics and Autonomous Systems*, volume 12, p. 187–198, 1994.

[15] H.J. Levesque. What is planning in the presence of sensing? In *AAAI-96*, volume 2, p. 1139–1145, Portland, Oregon, 1996.

[16] H.J. Levesque and R. Reiter. High-level robotic control: beyond planning. In: *Integrating Robotics Research: Taking the Next Big Leap*. Position paper. AAAI 1998 Spring Symposium, Stanford University, March 23-25, 1998. Available at `http://www.cs.toronto.edu/~cogrobo/`

[17] H.J. Levesque, R. Reiter, Y. Lespérance, F. Lin and R. Scherl. *Golog* : A logic programming language for dynamic domains. *J. of Logic Programming, Special Issue on Actions*, 1997, volume 31, N 1–3, p. 59–83.

[18] F. Lin and R. Reiter. State constraints revisited. In *J. of Logic and Computation, special issue on actions and processes*, 1994, volume 4, p. 655–678.

[19] J.McCarthy, P.Hayes Some philosophical problems from the standpoint of artificial intelligence. In: B.Meltzer and D.Michie (editors), *Machine Intelligence*, volume 4, p. 463–502, Edinburgh University Press, 1969.

[20] P.J. Ramadge and W.M. Wonham. The Control of Discrete Event Systems. Proceedings of the IEEE, 77(1): 81–98, 1989.

[21] R. Reiter The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression. In: Vladimir Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, p. 359–380. Academic Press, San Diego, CA, 1991.

[22] R. Reiter. KNOWLEDGE IN ACTION: Logical Foundations for Describing and Implementing Dynamical Systems. A draft of the first eight chapters of a book. Available at `http://www.cs.toronto.edu/~cogrobo/`

[23] R. Reiter. Sequential, temporal Golog. In *Principles of Knowledge Representation and Reasoning*. Proceedings of the 6th International Conference (KR'98), Trento, Italy, June 2-5, 1998, this volume.

[24] E. Sandewall. Logic-Based Modelling of Goal-Directed Behavior. In *Linköping Electronic Articles in Computer and Information Science*, volume 2, N 19, 1997. Available at: `http://www.ep.liu.se/ea/cis/1997/019/`

[25] M.J. Schoppers. Universal plans for reactive robots in unpredictable environments In *10th International Joint Conference on Artificial Intelligence (IJCAI-87)*, p. 1039–1046, 1987.

[26] M.J. Schoppers. In defense of reaction plans as caches. In *AI Magazine*, volume 10, N 4, p. 51–60, 1989.

[27] M.J. Schoppers. Building monitors to exploit open-loop and closed-loop dynamics. In Proceedings of the 1st *International Conference on Artificial Iintelligence Planning Systems (AIPS-92)*, June 15-17, College Park, Maryland, p. 204–213, 1992

[28] M. Soutchanski. Execution monitoring of high-level robot programs. University of Toronto, PhD thesis, forthcoming.

[29] P. Traverso, A. Cimatti, L. Spalazzi. Beyond the single planning paradigm: introspective planning. In *10th European Conference on Artificial Intelligence (ECAI-92)*, p. 643–647, Vienna, August 3-7, 1992.

[30] P. Traverso and L. Spalazzi. A Logic for Acting, Sensing and Planning. In *14th International Joint Conference on Artificial Intelligence*, volume 2, p. 1941–1949, 1995.

Building, Merging, and Revising Theories II

A general approach for inconsistency handling and merging information in prioritized knowledge bases

S. Benferhat[1] D. Dubois[1] J. Lang[1] H. Prade[1] A. Saffiotti[2] P. Smets[2]

[1] IRIT, Université Paul Sabatier
118 Route de Narbonne
Toulouse 31062 Cedex 4, France
{benferhat, dubois, lang, prade}@irit.fr

[2] IRIDIA, Université Libre de Bruxelles
50 av. F. Roosevelt
B-1050 Bruxelles, Belgium
{psmets, asaffio}@ulb.ac.be

Abstract

Several well-known prioritized methods for inconsistency-tolerant reasoning are based on the selection of one or several consistent subbases, and a formula is nontrivially inferred from the original inconsistent knowledge base if and only if this formula is a classical consequence of all the selected subbases. This paper unifies these methods in the framework of a symbolic set-valued label calculus based on a general partial preordering between sets, satisfying prescribed properties of cancellation and compatibility with inclusion that make it similar to a comparative probability. A prescribed label is assigned to each formula of the inconsistent possibly stratified knowledge base. Then we show how to turn a prioritized, or penalty-labelled inconsistent knowledge base into a "semantically equivalent" one (in the sense of the preservation of a partial ordering of interpretations) where inconsistency-tolerant inference can be computed using resolution-based refutation in the style of possibilistic logic. This equivalent labelled knowledge-base is obtained by a symmetric and associative syntactic combination of labelled formulas. The combination can thus process one formula after the other in an incremental fashion. This technique obviates the need for explicit enumeration and recalculation of maximal consistent subbases.

1. INTRODUCTION

In practice, knowledge bases are sometimes inconsistent, due to the presence of rules having exceptions, or because the available knowledge comes from several, not necessarily agreeing sources. One way of tackling inconsistency is to use the so-called coherence approaches [Nebel, 1991; Benferhat , et al., 1995a] which can be characterized by the two following steps: i) give up some formulas of the inconsistent knowledge base **K** in order to restore its consistency; this operation results into one or several consistent subbases of **K**; and ii) use classical entailment on these consistent subbases to defining plausible conclusions from the knowledge base **K**. We can distinguish between two classes of coherence theories: coherence theories based on the choice of one consistent subbase (not necessarily maximal), and coherence theories based on the selection of several maximal consistent subbases. When priorities attached to pieces of knowledge are available, the task of coping with inconsistency is simplified, because priorities give information about how to solve conflicts. The importance of priority in information systems has been first pointed out in [Fagin et al., 1983]. Examples of prioritized coherence approaches which select one consistent subbase are the possibilistic logic approach [Dubois et al., 1994], Williams' approach to belief revision [Williams, 1996]. Examples of approaches which use several consistent subbases are Rescher's notion of acceptable subbases [Rescher, 1976], Brewka's preferred sub-theories [Brewka, 1989], the lexicographical system [Benferhat et al., 1993], [Lehmann, 1995]. From a minimal change point of view, the second class seems more satisfactory since it keeps as many formulas as possible while the first class selects one consistent subbase which is often not maximal. However, from a pragmatic point of view, the first class based on a direct use of priorities is very interesting since its computational complexity is close to the one of classical logic, while selecting several maximal consistent subbases is computationally very heavy; see [Nebel 1994]. A part of the computational difficulties of the second class are often due to the fact that we take into account all the beliefs of a knowledge base **K** and compute the set of maximally consistent subsets of **K** although some beliefs are not used in inferences. For instance, consider the knowledge base composed of three formulas $\mathbf{K}=\{\alpha, \neg\alpha, \beta\}$ where all the beliefs are assumed to be equally reliable. This knowledge base is inconsistent and admits two maximally consistent subsets $A=\{\alpha, \beta\}$ and $B=\{\neg\alpha, \beta\}$. We can easily check that for any formula ψ, we have both $A\vdash\psi$ and $B\vdash\psi$ iff $\beta\vdash\psi$. This means that we can ignore the two beliefs $\{\alpha, \neg\alpha\}$ without changing the set of plausible consequences of **K** in the sense of maximal consistent subbases inference.

This paper pursues an investigation of inconsistent-tolerant reasoning started in [Benferhat et al., 1993] by proposing two new contributions. First, it offers a general framework to inconsistency-tolerant reasoning in prioritized knowledge bases that encompasses several well-known systems, like Brewka's preferred sub-theories or the lexicographical approach. Moreover, it presents a new way to address inconsistency-tolerant reasoning, viewing it as a data fusion problem aiming at merging the individual beliefs in the knowledge base. More precisely, we view each belief in the knowledge base **K** as a one-formula labelled knowledge base provided by some source. We propose to merge these one-formula labelled knowledge bases into a new labelled knowledge base (possibly containing some additional formulas having new levels). It generalizes to abstract labels a combination technique recently introduced in the possibilistic logic setting [Benferhat et al. 1997]. We show that this syntactic combination of labelled knowledge bases offers a method for tackling inconsistency. Namely, given a prioritized inconsistent knowledge **K**, we construct a labelled knowledge base $\mathbb{C}(\mathbf{K})$ such that any plausible conclusion inferred from **K** using some existing inconsistency-tolerant consequence relation, is also a labelled conclusion from $\mathbb{C}(\mathbf{K})$ in the sense of a partially-ordered version of possibilistic logic, and conversely. It is also shown that the same approach can also encompass inference in a penalty logic where the formulas are not simply prioritized, but are associated with a cost to pay if the formulas are not satisfied. We first introduce a general notion of labelled knowledge bases.

2. LABELLED KNOWLEDGE BASES AND PREFERENTIAL ENTAILMENT

In this paper, we only consider a finite propositional language. The symbol $\vdash$ represents the classical consequence relation, Greek letters $\alpha,\beta,\ldots$ represent formulas. We denote by Ω the set of all classical interpretations, and $\top$ denotes tautology. In the following, the symbol **K** denotes a prioritized knowledge base, where priorities are labels (as it is presented in this section), or a simple ranking between beliefs (see Section 3.1.), or also positive integer numbers associated to beliefs (see Section 3.1.).

Consider labelled knowledge bases having the following form: $\mathbf{K}=\{(\phi_i : I_i)\}$ where I_i is a non-empty set called the label of formula ϕ_i. Intuitively, the label I_i encodes information about the sources which provide the formulas (beliefs) ϕ_i. More precisely the pair $(\phi_i : I_i)$ means that at least one of the sources in I_i has provide the belief ϕ_i. In the following, we denote by $\mathcal{S}=\{s_1,\ldots,s_n\}$ a finite set of sources. A label I_i associated to ϕ_i can be a subset of sources or a subset of values attached to the involved sources. Some knowledge about the relative reliability of groups of sources is supposed to be known under the form of a preference relation on $2^{\mathcal{S}}$:

Definition 1: A preference relation on $2^{\mathcal{S}}$ is said to be a C-preference relation, denoted $\leq_{\mathcal{S}}$, if it satisfies the following conditions: Let I, J, $M\subseteq\mathcal{S}$,

A1. $\leq_{\mathcal{S}}$ is reflexive and transitive;

A2. $I\subsetneq J$ implies $I <_{\mathcal{S}} J$,

(where $\subsetneq$ denotes strict inclusion) and

A3. If I, J, M are disjoint subsets then :

$$I \cup J \leq_{\mathcal{S}} I \cup M \Leftrightarrow J \leq_{\mathcal{S}} M,$$

where : $I<_{\mathcal{S}}J$ means that $I\leq_{\mathcal{S}}J$ but not $J\leq_{\mathcal{S}}I$.

We define the equivalence relation $I=_{\mathcal{S}}J$ as $I\leq_{\mathcal{S}}J$ and $J\leq_{\mathcal{S}}I$. The equivalence relation should not be confused with the incomparability relation defined by if neither $I\leq_{\mathcal{S}}J$ nor $J\leq_{\mathcal{S}}I$ hold. Thus, $\leq_{\mathcal{S}}$ is a kind of partially ordered qualitative probability [Lehmann, 1996], where all non-empty events have a "non-null" probability value due to A2.

Given a family $\mathcal{B}$ of subsets of $\mathcal{S}$, we define its maximal elements in the usual way: $\mathrm{Max}(\mathcal{B}, \leq_{\mathcal{S}})=\{I_i \mid \nexists I_j\in \mathcal{B}$ such that $I_i<_{\mathcal{S}}I_j\}$. The minimum operator Min can be defined in a similar way. For the sake of simplicity, we simply write $\mathrm{Max}(\mathcal{B})$ instead of $\mathrm{Max}(\mathcal{B}, \leq_{\mathcal{S}})$. When $\mathrm{Max}(\mathcal{B})$ is singleton $\{I\}$, then sometimes we write $\mathrm{Max}(\mathcal{B}) = I$.

In the following, we consider problems defined by pairs $(\mathbf{K}, \mathcal{C})$ (the available data), where **K** is a labelled knowledge base $\mathbf{K}=\{(\phi_i : I_i)\}$ and $\mathcal{C}$ is a set of constraints on $2^{\mathcal{S}}$, encoding some partial information about the preference order $\leq_{\mathcal{S}}$. All labels I_i in **K** are different and are singletons. The $\leq_{\mathcal{S}}$ relation partially defined by $\mathcal{C}$ then has to be closed with respect to axioms A1-A3 above. Our aim is to deal with the inconsistencies of **K**, using the relation $\leq_{\mathcal{S}}$. Roughly speaking, the relation $<_{\mathcal{S}}$ helps selecting a subset of **K** to be removed in order to recover the consistency of **K**. $I<_{\mathcal{S}}J$ means that, in order to restore the consistency of **K**, we prefer to remove the set of beliefs supported by the sources in I rather than to remove the set of beliefs supported by the sources in J. In this way, the condition A2 becomes very natural since it corresponds to the idea of minimal change : only a smaller subset necessary to restore the consistency of **K** should be removed. Condition A3 simply means that only the beliefs which differ between J and M should be taken into account to distinguish between J and M. A1-A3 are general requirements for $\leq_{\mathcal{S}}$.

The $<_{\mathcal{S}}$ partial order can also be read as expressing a negligibility relation. More specifically, if we assume that the sources of information in $\mathcal{S}$ can make mistakes (exceptions), then we can read $I <_{\mathcal{S}} J$ as saying that the

likelihood that some of the sources in J is wrong is negligible with respect to the likelihood that some source in I is wrong. Axioms A2 and A3 then have a natural reading: A2 says that labels that need more sources are less likely to be wrong; and A3 says that uniformly adding new sources does not modify the relative reliability of labels.
Note that from A1-A3 we get: (A'2) $I \subseteq J$ implies $I \leq_{\mathcal{S}} J$. We also have for any $I \neq \emptyset$, $I <_{\mathcal{S}} \mathcal{S}$ and $\emptyset <_{\mathcal{S}} I$.

Definition 2: Let $\mathcal{C}$ be a set of constraints on $2^{\mathcal{S}}$ of the form $I<J$ or $I \leq J$. A closure of $\mathcal{C}$ is the smallest relation, denoted by $\leq_{\mathcal{S}}$, on $2^{\mathcal{S}}$ satisfying A1-A3 and containing $\mathcal{C}$.

Proposition 1: When $\mathcal{C}$ is consistent with the axioms A1-A3 then the closure of $\mathcal{C}$ is always unique.
Proof (sketch):
Indeed, let $R_{\mathcal{C}}=\{(I,J);\ I \leq J \in \mathcal{C}$ or $I<J \in \mathcal{C}\}$ and $R_C=\{(I, J);\ I \subseteq J\}$. Now, complete $R_{\mathcal{C}} \cup R_C$ by considering $R = \{(I \cup M,\ J \cup M);\ \forall I, J, M$ disjoint subsets, $(I,J) \in R_{\mathcal{C}} \cup R_C\}$. Finally, take the transitive closure of R. This is the closure of $\mathcal{C}$ in the sense of Definition 2. Note that the consistency of $\mathcal{C}$ with the axioms A1-A3 should be checked at each step of the completion process. ■

In the following, we assume that $\mathcal{C}$ is always consistent with the axioms A1-A3.

Example 1 (continued):
Let $\mathcal{C}=\{\{s_2,s_3\}<\{s_1\},\ \{s_2\} \leq \{s_3\},\ \{s_3\} \leq \{s_2\},\ \{s_1\}<\{s_4\}\}$. The closure $\leq_{\mathcal{S}}$ of $\mathcal{C}$ is pictured in the following figure (reflexivity and transitivity are not represented for clarity reason):

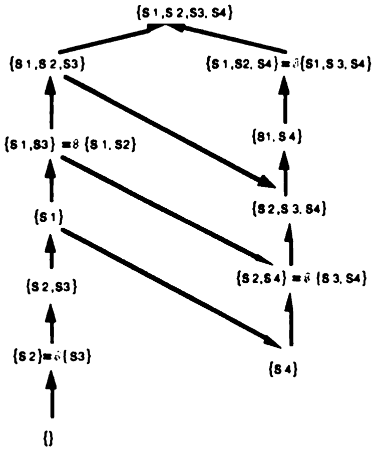


Figure 1
For instance, $\{s_2, s_3, s_4\} <_{\mathcal{S}} \{s_1, s_4\}$ is obtained by applying A3 to $\{s_2, s_3\} <_{\mathcal{S}} \{s_1\}$ which is in $\mathcal{C}$. ■

From a labelled knowledge base **K** and the ordering $\leq_{\mathcal{S}}$, two preferential nonmonotonic entailment relations, denoted by $\models_{L,\mathbf{K}}$ and $\models_{\kappa,\mathbf{K}}$, can be defined in the spirit of Shoham (1988). We denote:

$$A_{\mathbf{K}}(\omega) = \{I_i \mid (\phi_i : I_i) \in \mathbf{K} \text{ and } \omega \models \neg\phi_i\}$$

the set of labels of formulas falsified by ω.

Consider two nonmonotonic entailment relations based on exploiting the sets $A_{\mathbf{K}}(\omega)$. Given $A_{\mathbf{K}}(\omega)$, define first the set of sources $L_{\mathbf{K}}(\omega)$ which claim that ω is impossible, namely:

$$L_{\mathbf{K}}(\omega) = \cup_{I_i \in A_{\mathbf{K}}(\omega)} I_i ,$$

$L_{\mathbf{K}}(\omega)$ is called the label of ω. Moreover, we can also select in $A_{\mathbf{K}}(\omega)$ the greatest labels in the sense of the relation $\leq_{\mathcal{S}}$:

$$\kappa_{\mathbf{K}}(\omega) = \text{Max}\ (A_{\mathbf{K}}(\omega)).$$

$\kappa_{\mathbf{K}}(\omega)$ is the set of non-dominated (i.e., the most important) labels of formula in **K** falsifying ω.

Clearly, the closer $L_{\mathbf{K}}(\omega)$ to being minimal according to $\leq_{\mathcal{S}}$, the more preferred is ω. The same applies with $\kappa_{\mathbf{K}}(\omega)$ provided that we lift the preference relation $\leq_{\mathcal{S}}$ to subsets of labels. More formally:

Definition 3: • An interpretation ω is said to be *L-preferred to* ω' if $L_{\mathbf{K}}(\omega) <_{\mathcal{S}} L_{\mathbf{K}}(\omega')$, and similarly
• An interpretation ω is said to be *κ-preferred to* ω' if $\forall I \in \kappa_{\mathbf{K}}(\omega')$, $\exists J \in \kappa_{\mathbf{K}}(\omega)$, such that $J <_{\mathcal{S}} I$.

Definition 4: An interpretation ω is said to be an *L-preferred (resp. κ-preferred) model* of a formula ϕ iff $\omega \models \phi$ and $\nexists \omega' \models \phi$ such that ω' is L-preferred (resp. κ-preferred) to ω.

Then, the entailment is defined in the following way:

Definition 5: $\alpha \models_{L,\mathbf{K}} \beta$ (resp. $\alpha \models_{\kappa,\mathbf{K}} \beta$) iff each L-preferred (resp. κ-preferred) model of α in the sense of the label-driven partial ordering $<_{\mathcal{S}}$ satisfies β.

The entailment $\models_{L,\mathbf{K}}$ will be called "labelled inference", while $\models_{\kappa,\mathbf{K}}$ will be called κ-semantic consequence. The κ−semantics generalizes the semantic basis of possibilistic logic [Dubois et al., 1994], System Z+ [Goldszmidt, and Pearl, 1991] (based on Spohn's calculus [1988]) and goes back to [Rescher, 1976].

Example 1 (continued):
Let $\mathbf{K} = \{\ (a : \{s_1\}), (\neg a \vee b : \{s_2\}),$
$(\neg a \vee \neg b: \{s_3\}), (\neg a \wedge c: \{s_4\})\}$, and
$\mathcal{C}=\{\{s_2,s_3\}<\{s_1\},\ \{s_2\} \leq \{s_3\},\ \{s_3\} \leq \{s_2\},\ \{s_1\}<\{s_4\}\}$.
Let: $\Omega = \{\ \omega_0 : \neg a \wedge \neg b \wedge \neg c,\ \omega_1 : \neg a \wedge \neg b \wedge c,$
$\omega_2 : \neg a \wedge b \wedge \neg c,\ \omega_3 : \neg a \wedge b \wedge c,\ \omega_4 : a \wedge \neg b \wedge \neg c,$
$\omega_5 : a \wedge \neg b \wedge c,\ \omega_6 : a \wedge b \wedge \neg c,\ \omega_7 : a \wedge b \wedge c\}$,
be the set of all interpretations. We have:
$A(\omega_0)=A(\omega_2)=\{\{s_1\},\{s_4\}\},\ \ A(\omega_1)=A(\omega_3)=\{\{s_1\}\},$

$A(\omega_4)=A(\omega_5)=\{\{s_2\},\{s_4\}\}$, $A(\omega_6)=A(\omega_7)=\{\{s_3\},\{s_4\}\}$.
Then we get:
$L_K(\omega_0)=L_K(\omega_2)=\{s_1, s_4\}$, $L_K(\omega_1)=L_K(\omega_3)=\{s_1\}$,
$L_K(\omega_4)=L_K(\omega_5)=\{s_2, s_4\}$, $L_K(\omega_6)=L_K(\omega_7)=\{s_3,s_4\}$,
and using Figure 1, we get:
$\kappa_K(\omega_0)=\kappa_K(\omega_2)=\text{Max }\{\{s_1\},\{s_4\}\} = \{s_4\}$
$\kappa_K(\omega_1)=\kappa_K(\omega_3)=\{s_1\}$,
$\kappa_K(\omega_4)=\kappa_K(\omega_5)=\text{Max }\{\{s_2\},\{s_4\}\} = \{s_4\}$
$\kappa_K(\omega_6)=\kappa_K(\omega_7)=\text{Max }\{\{s_3\},\{s_4\}\} = \{s_4\}$
The two following figures summarize the relationships between elements of Ω using respectively the function L and κ:

$$\begin{tikzcd}
L(\omega_1) =_{\mathcal{E}} L(\omega_3) \arrow[d] \\
L(\omega_4) =_{\mathcal{E}} L(\omega_5) =_{\mathcal{E}} L(\omega_6) =_{\mathcal{E}} L(\omega_7) \arrow[d] \\
L(\omega_0) =_{\mathcal{E}} L(\omega_2)
\end{tikzcd}$$

Figure 2

$$\begin{tikzcd}
\kappa(\omega_1) =_{\mathcal{E}} \kappa(\omega_3) \arrow[d] \\
\kappa(\omega_0) =_{\mathcal{E}} \kappa(\omega_2) =_{\mathcal{E}} \kappa(\omega_4) =_{\mathcal{E}} \kappa(\omega_5) =_{\mathcal{E}} \kappa(\omega_6) =_{\mathcal{E}} \kappa(\omega_7)
\end{tikzcd}$$

Figure 3

Note that we have: $T \models_{L,K} \neg a \wedge c$ and $T \models_{\kappa,K} \neg a \wedge c$, but $\neg b \wedge \neg c \models_{L,K} a$ while $\neg b \wedge \neg c \not\models_{\kappa,K} a$. ■

3. ENCODING COHERENCE THEORIES

In this section we show that some of the well-known coherence theories mentioned in the introduction can be recovered using the labelled inference described above by adding some specific conditions. First, we recall these well-known inference proposals.

3.1. Reminder on coherence theories

Brewka's preferred sub-theories [Brewka, 1989] and the lexicographical inference [Benferhat et al., 1993] [Lehmann, 1995] deal with layered inconsistent knowledge bases of the form $\mathbf{K}=S_1\cup\ldots\cup S_n$, such that formulas in S_i have the same level of priority (or certainty) and are more reliable than the ones in S_j where j<i. Formulas of S_n are the most important beliefs and those in S_1 are the least important ones. We denote subbases by capital letters A,B,...also represented in a stratified way, namely $A=A_1\cup\ldots\cup A_n$ where $\forall j=1,n$, $A_j \subseteq S_j$. There exist two criteria for defining such preferred subbases of **K**: set-inclusion, or cardinality [Benferhat et al., 1993].

Definition 6: A consistent subbase $A=A_1\cup\ldots\cup A_n$ is an *inclusion-preferred subbase* of **K** iff there is no consistent subbase $B = B_1\cup\ldots\cup B_n$ of **K** such that:

$\exists i \leq n$, where $A_i \subsetneq B_i$ and for $j > i$, $B_j = A_j$.

Definition 7: A consistent subbase $A=A_1\cup\ldots\cup A_n$ is a *cardinality preferred subbase* (also called *lexicographically preferred subbase*) of **K** iff there is no consistent subbase $B=B_1\cup\ldots\cup B_n$ such that: $\exists i \leq n$, where $|B_i| > |A_i|$ and for $j > i$, $|B_j| = |A_j|$ where $|A|$ is the cardinality of A.

Selecting a subset of the set of maximally consistent subbases of **K** using a cardinality criterion is of interest in diagnostic problems for instance [De Kleer, 1990].

In penalty logic [Pinkas and Loui, 1992], [Dupin et al., 1994] each piece of information ϕ_i is associated with the penalty $c_i \in N$, read as the price to pay if ϕ is not satisfied. Let $\mathbf{K} = \{(\phi_i\ c_i)/\ i=1,n\}$ be a penalty knowledge base.

Definition 8: A consistent subbase A of **K** is a *penalty preferred subbase* of **K** iff there is no consistent subbase B of **K** such that: $\sum\{c_i \mid (\phi_i\ c_i)\in \mathbf{K}\text{-}A\} < \sum\{c_j \mid (\phi_j\ c_j)\in \mathbf{K}\text{-}B\}$.

Definition 9: A formula ϕ is said to be a *Incl-consequence* (resp.*Lex-consequence and Penalty-consequence*) of **K** in the context α, denoted by $\alpha \vdash_{\text{Incl},\mathbf{K}} \phi$ (resp.$\alpha \vdash_{\text{lex},\mathbf{K}} \phi$ and $\alpha \vdash_{\text{pen},\mathbf{K}} \phi$), iff it is entailed classically from each inclusion-preferred subbase of **K** (resp. cardinality-preferred and penalty preferred) which is consistent with α.

Remark: If summations are changed into maximum in Definition 8, the c_i's turned into certainty degrees valued in [0,1] and reversing the inequality, Definition 9 yields possibilistic logic [Dubois et al., 1994]. This paper exploits the parallel with possibilistic logic, especially in Section 4.2., when defining the syntactic inference machinery on the knowledge base obtained after the combination step.

3.2. Capturing coherence theories by means of labelled entailment

In this section, given a stratified knowledge base $\mathbf{K}=S_1\cup\ldots\cup S_n$ or a penalty knowledge base $\mathbf{K}=\{(\phi_i\ c_i)/i=1,\ldots,n\}$, we construct a labelled knowledge base **K'** and a preference ordering $\leq_{\mathcal{G}}$ such that: $\alpha \models_{x,\mathbf{K}} \beta$ iff $\alpha \models_{L,\mathbf{K}'} \beta$ where $x \in \{\text{pen; lex; incl}\}$.

First consider the lexicographic system. Let us give the transformation. The labelled knowledge base associated to $\mathbf{K}=S_1\cup\ldots\cup S_n$ is defined by $\mathbf{K}_{\mathbf{lex}} = \{(\phi_{ij} : s_{ij})\}$ where ϕ_{ij} be the j-th belief (according to an arbitrary numbering) in the i-th layer S_i. The set of sources is defined by: $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2 \cup \ldots \cup \mathcal{S}_n$ such that $\mathcal{S}_i = \{s_{i1} \ldots, s_{i|S_i|}\}$ where $|S_i|$ is the number of beliefs in S_i. Now, let us define the preference relation $\leq_{\mathcal{S}}$ as follows:

i) for i=1,n and for any j,k=1,...,$|S_i|$,
$$\{s_{ij}\} =_{\mathcal{S}} \{s_{ik}\},$$

ii) For i=2,n, and for j=1,...,$|S_i|$,
$$\cup_{k=1,i-1}\, \mathcal{S}_k <_{\mathcal{S}} \{s_{ij}\}.$$

Of course, $\leq_{\mathcal{S}}$ should be completed with the properties A1-A3. In particular, from A2 we get $\{s_{jm}\} <_{\mathcal{S}} \{s_{ik}\}$ for i>j. Intuitively, the second constraint means that to restore the consistency of **K**, we prefer to remove all the beliefs having a rank strictly lower than i rather than removing one belief of rank i.

Proposition 2: $\alpha \models_{lex,\mathbf{K}} \beta$ iff $\alpha \models_{L,\mathbf{K}_{lex}} \beta$.

The proof is given in the appendix.

The encoding of preferred sub-theories is very similar to the lexicographic one. We define $\mathbf{K}_{\mathbf{Incl}}$ and $\mathcal{S}$ in the same way and the preference relation $\leq_{\mathcal{S}}$ is defined by means of condition ii) above, only, as follows:

- For i=2,n, and for j=1,...,$|S_i|$,
$$\cup_{k=1,i-1}\, \mathcal{S}_k <_{\mathcal{S}} \{s_{ij}\}.$$

Note that for $j \neq k$ we do not specify the relationships between $\{s_{ij}\}$ and $\{s_{ik}\}$, they remain incomparable (and not equivalent as in the lexicographic case).

Proposition 3: $\alpha \models_{\mathrm{Incl},\mathbf{K}} \beta$ iff $\alpha \models_{L,\mathbf{K}_{\mathbf{Incl}}} \beta$.

The encoding of penalty logic is straightforward. Let $\mathbf{K}=\{(\phi_i\ c_i)/\ i=1,n\}$ be a penalty knowledge base. The labelled knowledge base is: $\mathbf{K}_{\mathbf{Pen}}=\{(\phi_i : \{c_i\})\}$ where ϕ_i is the i-th belief (according to an arbitrary enumeration) in **K**. The set of sources is $\mathcal{S}=\{s1,\ldots,sn\}$, and the relation $\leq_{\mathcal{S}}$ is defined as follows: $I <_{\mathcal{S}} J$ iff $\sum_{i\in I} c_i < \sum_{j\in J} c_j$.

Proposition 4: $\alpha \models_{pen,\mathbf{K}} \beta$ iff $\alpha \models_{L,\mathbf{K}_{\mathbf{Pen}}} \beta$.

4. COMPILING INCONSISTENT LABELLED KNOWLEDGE BASES

The coherence approaches have two major drawbacks when coping with inconsistencies:

- Computing the set of preferred consistent subsets of **K** exploits all the beliefs of a knowledge base **K** even if some beliefs are not useful to make inferences. An extreme case (which is of course non-realistic) is described by the following situation. Let us assume that the set of propositional symbols in the language is $\{p_1,\ldots,p_n\}$. Let $\mathbf{K}=S_1\cup\ldots\cup S_n$ be a stratified knowledge base where $S_i=\{\neg p_i\ ,\ p_i\}$. Then we can easily check that there are exactly 2^n cardinality preferred subbases (resp. inclusion-preferred). However, we can also check that only tautologies are lex-consequences of **K** (resp. Incl-consequences). We will see later (Example 2) that by using our syntactic combination, we simply get one labelled formula $(\perp : I)$, from which only tautologies can be considered as plausible conclusions.

- When computing preferred sub-bases incrementally, it is not possible to forget the removed beliefs because they can reappear later, as shown on this example: let $\mathbf{K}=S_1\cup S_2\cup S_3\cup S_4$ such that $S_4=\{\alpha\}$, $S_3=\{\delta\}$, $S_2=\{\neg\delta\wedge\xi\}$ and $S_1=\{\beta, \neg\beta\vee\neg\delta\}$. We can easily check that there are only two cardinality preferred sub-bases which are $A=\{\alpha\}\cup\{\delta\}\cup\{\neg\beta\vee\neg\delta\}$ and $B=\{\alpha\}\cup\{\delta\}\cup\{\beta\}$. Now if a new belief$\{\neg\delta\wedge\neg\beta\}$ is entered in S_4 then we only get one preferred sub-base $A=\{\alpha,\neg\delta\wedge\neg\beta\}\cup\{\neg\delta\wedge\xi\}\cup\{\neg\beta\vee\neg\delta\}$ which cannot be recovered from the two previously preferred subbases.

In this section, we propose an alternative way for implementing the coherence theories using a syntactic combination, which avoids the two previous limitations. This can be described in two steps:

- The first step consists in viewing the problem of inconsistencies as a data fusion problem. More precisely, we view each belief $(\phi_i : I_i)$ of **K** as a one-formula knowledge base $\mathbf{K}_i=\{(\phi_i : I_i)\}$ provided by some source. Then, we merge these one-formula knowledge bases $\mathbf{K}_i$ using a binary combination operator $\Diamond$. This operation is mainly the qualitative counterpart of the ones proposed in (Boldrin and Sossai, 1995) and (Benferhat et al., 1997). The result of merging these knowledge bases is a new knowledge base that we denote by $\mathbb{C}(\mathbf{K})$. This new knowledge base $\mathbb{C}(\mathbf{K})$, called the compilation of K, should be semantically equivalent in the sense that:
$$\alpha \models_{L,\mathbf{K}} \beta \quad \text{iff} \quad \alpha \models_{\kappa,\mathbb{C}(\mathbf{K})} \beta$$

To obtain this equivalence, it is enough that the following equality holds:
$$\forall \omega \in \Omega, \quad \kappa_{\mathbb{C}(\mathbf{K})}(\omega) = L_{\mathbf{K}}(\omega).$$

We shall check that $\kappa_{\mathbb{C}(\mathbf{K})}(\omega)$ is always a singleton. This first step is then summarized by the following figure:

$$\begin{tikzcd}
\mathbf{K} \arrow[r, "\Diamond"] \arrow[d, "L"] & \mathbb{C}(\mathbf{K}) \arrow[d, "\kappa"] \\
\alpha \models_{L,\mathbf{K}} \beta \arrow[r, phantom, "\text{iff}"] & \alpha \models_{\kappa,\mathbb{C}(\mathbf{K})} \beta
\end{tikzcd}$$

Figure 4

• The second step consists in applying a syntactic efficient inference $\vdash_\kappa$ which is complete and sound with the κ–semantics, namely:

$$\mathbb{C}(\mathbf{K}) \cup \{(\alpha : \mathcal{S})\} \vdash_\kappa \beta \text{ iff } \alpha \models_{\kappa,\mathbb{C}(\mathbf{K})} \beta.$$

α is a new certain fact, therefore it is labelled with the highest rank, namely $\mathcal{S}$.

We will show that the syntactic entailment $\vdash_\kappa$ is in fact based on selecting a single consistent subset of $\mathbb{C}(\mathbf{K}) \cup \{(\alpha : \mathcal{S})\}$, and making a classical inference step from this subbase. The entailment $\vdash_\kappa$ coincides with the possibilistic logic inference when the relation $\leq_{\mathcal{S}}$ is complete. Noticeably, the selected subbase remains unique even if the relation $\leq_{\mathcal{S}}$ is not complete.

These two steps are described in the two following sections.

4.1. Compiling K

In this section, we propose to compile the inconsistent knowledge base into a new semantically equivalent one, denoted $\mathbb{C}(\mathbf{K})$. This is obtained by applying an associative binary combination operator, denoted by $\Diamond$, to one-formula knowledge bases $\mathbf{K}_i=\{\phi_i\}$ (with $\phi_i \in \mathbf{K}$). $\mathbb{C}(\mathbf{K})$ will be obtained by applying the combination operator $\Diamond$ iteratively. $\mathbb{C}(\mathbf{K})$ should be such that: $\forall \omega \in \Omega$,

$$L_{\mathbf{K}}(\omega) = \kappa_{\mathbb{C}(\mathbf{K})}(\omega)$$
$$= \text{Max}\{I_p \,/\, (\phi_p : I_p) \in \mathbb{C}(\mathbf{K}) \text{ and } \omega \models \neg\phi_p\},$$

where its maximum is unique.

By convention if ω satisfies all the beliefs in $\mathbb{C}(\mathbf{K})$ then $\kappa_{\mathbb{C}(\mathbf{K})}(\omega)=\emptyset$. Such $\mathbb{C}(\mathbf{K})$ is not unique, we only exhibit a simple one.

Let us first consider the simple case where $\mathbf{K}$ contains two beliefs $\mathbf{K}=\{(\phi : I), (\psi : J)\}$. Let us exhibit what is the labelled knowledge base corresponding to the combination of two one-formula labelled knowledge bases $\mathbf{K}_1 = \{(\phi : I)\}$ and $\mathbf{K}_2=\{(\psi : J)\}$.

Definition 10: The combination of $\mathbf{K}_1$ and $\mathbf{K}_2$ is the three labelled formulas:

$$\mathbf{K}_1 \Diamond \mathbf{K}_2 = \{(\phi : I), (\psi : J), (\phi \vee \psi : I \cup J)\}.$$

Proposition 5: If $\mathbb{C}(\mathbf{K}) = \mathbf{K}_1 \Diamond \mathbf{K}_2$, then the equivalence $\alpha \models_{L,\mathbf{K}} \beta$ iff $\alpha \models_{\kappa,\mathbb{C}(\mathbf{K})} \beta$ is satisfied.

<u>Proof:</u> This can be easily checked. Indeed, there are four different cases to consider:

• If $\omega \models \phi \wedge \psi$ then $\kappa_{K1 \Diamond K2}(\omega) = \emptyset = L_{\mathbf{K}}(\omega)$.

• If $\omega \models \phi \wedge \neg\psi$ then: $\kappa_{K1 \Diamond K2}(\omega) = J = L_{\mathbf{K}}(\omega)$

• By symmetry; If $\omega \models \neg\phi \wedge \psi$ then:

$$\kappa_{K1 \Diamond K2}(\omega) = I = L_{\mathbf{K}}(\omega)$$

• If $\omega \not\models \phi$ and $\omega \not\models \psi$ then this implies that $\omega \not\models \phi \vee \psi$. Hence using A2:

$$\kappa_{K1 \Diamond K2}(\omega) = \text{Max}\{I, J, I \cup J\} = I \cup J = L_{\mathbf{K}}(\omega). \quad \blacksquare$$

For instance if $\mathbf{K}_1=\{(\alpha: \{1\})\}$; $\mathbf{K}_2=\{(\beta: \{2\})\}$ then $\mathbf{K}_1 \Diamond \mathbf{K}_2 = \{(\alpha: \{1\}), (\beta: \{2\}), (\alpha \vee \beta : \{1, 2\})\}$. In the penalty case if $\mathbf{K}_1 = \{(\alpha, a)\}$ and $\mathbf{K}_2 = \{(\beta, b)\}$ where a and b are positive numbers, then $\mathbf{K}_1 \Diamond \mathbf{K}_2 = \{(\alpha: \{a\}), (\beta: \{b\}), (\alpha \vee \beta : \{a, b\})\}$ that will be semantically equivalent to $\mathbf{K} = \{(\alpha, a), (\beta, b), (\alpha \vee \beta, a+b)\}$.

Note that the combination step creates from $(\phi : I)$, $(\psi : J)$ a third formula $(\phi \vee \psi : I \cup J)$ with $I <_{\mathcal{S}} I \cup J$ and $J <_{\mathcal{S}} I \cup J$ (from A2). This expresses a reinforcement. This is intuitively satisfying since even if ϕ and ψ conflict, $\phi \vee \psi$ is a minimal basis of agreement between the sources. Moreover, even if ϕ and ψ are consistent, it may happen that ϕ conflicts with an input fact α; then using $\phi \vee \psi$ may yield a conclusion which would not be possible from $(\phi : I)$, $(\psi : J)$ only in the case when $J <_{\mathcal{S}} I$ (using the κ-semantics).

The previous transformation has characterized the combination of two labelled knowledge bases containing exactly one formula. Let us generalize to the case where $\mathbf{K}$ contains n beliefs. This is carried out using the following proposition. Let $\mathbf{K} = \{(\phi_i : I_i)/\ i=1,n\}$ be a labelled knowledge base. Let $\mathbf{K}_i=\{(\phi_i : I_i)\}$ be n one-formula labelled knowledge bases. Let us assume that $\mathbf{K}'$ is the result of combining the first $j \geq 1$ one-formula knowledge bases $\mathbf{K}_{i=1,j}$, and $\mathbf{K}''$ is the result of combining the remaining (n-j) knowledge bases $\mathbf{K}_{i=j+1,n}$. Namely we assume that:

$$L_{\cup_{i=1,j}\mathbf{K}_i}(\omega) = \kappa_{\mathbf{K}'}(\omega);$$
$$L_{\cup_{i=j+1,n}\mathbf{K}_i}(\omega) = \kappa_{\mathbf{K}''}(\omega)$$

Clearly, we have:

$$L_{\mathbf{K}}(\omega) = L_{\cup_{i=1,j}\mathbf{K}_i}(\omega) \cup L_{\cup_{i=j+1,n}\mathbf{K}_i}(\omega).$$

Definition 11: Let $\mathbf{K}' \vee \mathbf{K}'' = \{(\phi_k \vee \psi_m : I_k \cup I_m) \mid \text{for } (\phi_k : I_k) \in \mathbf{K}' \text{ and } (\psi_m : I_m) \in \mathbf{K}''\}$ be the Cartesian union of $\mathbf{K}'$ and $\mathbf{K}''$. Then, we define the syntactic combination of $\mathbf{K}'$ and $\mathbf{K}''$ as equal to:

$$\mathbf{K}' \Diamond \mathbf{K}'' = \mathbf{K}' \cup \mathbf{K}'' \cup (\mathbf{K}' \vee \mathbf{K}'').$$

Proposition 6: If $\mathbb{C}(\mathbf{K}) = \mathbf{K}' \Diamond \mathbf{K}''$ then the equivalence $\alpha \models_{L,\mathbf{K}} \beta$ iff $\alpha \models_{\kappa,\mathbb{C}(\mathbf{K})} \beta$ is satisfied.

The proof is given in the appendix.

Proposition 7: The syntactic combination $\Diamond$ is commutative and associative, namely:

i) $\mathbf{K}_1 \Diamond \mathbf{K}_2 = \mathbf{K}_2 \Diamond \mathbf{K}_1$.

ii) $(\mathbf{K}_1 \Diamond \mathbf{K}_2) \Diamond \mathbf{K}_3 = \mathbf{K}_1 \Diamond (\mathbf{K}_2 \Diamond \mathbf{K}_3)$.

The previous proposition is important, since it means that it does not matter the order in which we combine the one-formula labelled knowledge bases, and that we can build $\mathfrak{C}(\mathbf{K})$ by merging single formulas one after the other.

Definition 12: The compilation of **K** is defined by:

$$\mathfrak{C}(\mathbf{K}) = \mathbf{K}_1 \Diamond \mathbf{K}_2 \Diamond \ldots \Diamond \mathbf{K}_{n-1} \Diamond \mathbf{K}_n.$$

Corollary: Let $\mathfrak{C}(\mathbf{K})$ the labelled knowledge base given by the previous definition. Then:

$$\forall\omega, \quad \kappa_{\mathfrak{C}(\mathbf{K})}(\omega) = L_{\mathbf{K}}(\omega)$$

In order to reduce the set of added beliefs in the syntactic combination, the following definitions and propositions are useful:

Definition 13: A labelled formula $(\phi : I) \in \mathbf{K}$ is said to be subsumed by **K** iff $\text{Filter}(\mathbf{K},(\phi{:}I)) \vdash \phi$ where $\text{Filter}(\mathbf{K},(\phi{:}I)) = \{\psi \,/\, (\psi : J) \in \mathbf{K} - \{(\phi : I)\} \text{ and } I \leq_{\mathcal{G}} J\}$.

Proposition 8: Let $(\phi : I) \in \mathbf{K}$ be subsumed. Then **K** and $\mathbf{K} - \{(\phi : I)\}$ are equivalent, i.e., they induce the same κ-ranking on Ω, namely:

$$\forall\omega\in\Omega,\ \kappa_{\mathbf{K}}(\omega) = \kappa_{\mathbf{K}-\{(\phi\,:\,I)\}}(\omega).$$

Proposition 9: The two labelled knowledge bases **K** and $\mathbf{K} - \{(\top : I)\}$, where $\top$ denotes tautology, are equivalent, i.e., they induce the same κ-ranking on Ω.

Proposition 10: Let $(\phi_1 : I_1) \in \mathbf{K}$ and $(\phi_2 : I_2) \in \mathbf{K}$ such that $I_1 =_{\mathcal{G}} I_2$. Let **K'** be a new labelled knowledge base constructed from **K** by replacing the two formulas by the unique labelled formula $(\phi_1 \wedge \phi_2 : I_1)$. Then **K'** and **K** induce the same κ-ranking on Ω.

On the basis of the previous propositions, we can estimate the size of the set of added formulas when combining two labelled knowledge bases $\mathbf{K}_1$ and $\mathbf{K}_2$ (i.e., the size of $\mathbf{K}_1 \vee \mathbf{K}_2$). The two following steps must be achieved before the combination:

. For each labelled knowledge base $\mathbf{K}_{i=1,2}$, remove the subsumed beliefs;

. In each labelled knowledge base $\mathbf{K}_{i=1,2}$, put formulas having the same rank in a unique formula (take their conjunction);

The knowledge bases resulting from the two above steps are denoted by $\mathbf{K'}_1$ and $\mathbf{K'}_2$.

Proposition 11: Let $n = |\{I_i \,/\, (\phi_i : I_i) \in \mathbf{K'}_1\}|$ and $m = |\{I_j \,/\, (\phi_j : I_j) \in \mathbf{K'}_2\}|$. Then combining $\mathbf{K}_1$ and $\mathbf{K}_2$ leads to add to $\mathbf{K}_1$ and $\mathbf{K}_2$ at most n*m beliefs.

In the extreme cases, we add n*m beliefs but this number is not always reached especially when the concatenation of $\mathbf{K}_1$ and $\mathbf{K}_2$ is highly inconsistent.

Example 1 (continued):

Let $\mathbf{K} = \{(a : \{s_1\}), (\neg a \vee b : \{s_2\}), (\neg a \vee \neg b : \{s_3\}), (\neg a \wedge c : \{s_4\})\}$, and $\mathcal{G} = \{\{s_2, s_3\} < \{s_1\}, \{s_2\} \leq \{s_3\}, \{s_3\} \leq \{s_2\}, \{s_1\} < \{s_4\}\}$.

Let $\mathbf{K}_1 = \{(a : \{s_1\})\}$, $\mathbf{K}_2 = \{(\neg a \vee b : \{s_2\})\}$
$\mathbf{K}_3 = \{(\neg a \vee \neg b : \{s_3\})\}$, $\mathbf{K}_4 = \{(\neg a \wedge c : \{s_4\})\}$

We have:

$$\mathbf{K}_1 \Diamond \mathbf{K}_2 = \mathbf{K}_1 \cup \mathbf{K}_2 \cup \{(\top : \{s_1, s_2\})\} = \mathbf{K}_1 \cup \mathbf{K}_2 \text{ (using Proposition 9)}$$

$$\mathbf{K}_1 \Diamond \mathbf{K}_2 \Diamond \mathbf{K}_3 = \mathbf{K}_1 \cup \mathbf{K}_2 \cup \mathbf{K}_3 \cup \{(\top : \{s_1, s_3\})\} \cup \{(\top : \{s_2, s_3\})\} = \mathbf{K}_1 \cup \mathbf{K}_2 \cup \mathbf{K}_3 \text{ (using Proposition 9)}$$

finally:

$$\mathfrak{C}(\mathbf{K}) = \mathbf{K}_1 \cup \mathbf{K}_2 \cup \mathbf{K}_3 \cup \mathbf{K}_4 \cup \{(a \vee c : \{s_1, s_4\})\} \cup \{(\neg a \vee b : \{s_2, s_4\})\} \cup \{(\neg a \vee \neg b : \{s_3, s_4\})\}$$

$$\equiv \{(a : \{s_1\}), (a \vee c : \{s_1, s_4\})\}, (\neg a \vee b : \{s_2, s_4\}), (\neg a \vee \neg b : \{s_3, s_4\}).$$

(using Proposition 8, since $(\neg a \vee b : \{s_2\})$, $(\neg a \vee \neg b : \{s_3\})$ and $(\neg a \wedge c : \{s_4\})$ are subsumed).

It is easy to check that : $L_{\mathbf{K}}(\omega) = \kappa_{\mathfrak{C}(\mathbf{K})}(\omega)$.

Indeed, we have:

$\kappa_{\mathfrak{C}(\mathbf{K})}(\omega_0) = \kappa_{\mathfrak{C}(\mathbf{K})}(\omega_2) = \text{Max}\ \{\{s_1\}, \{s_1, s_4\}\} = \{s_1, s_4\}$

$\kappa_{\mathfrak{C}(\mathbf{K})}(\omega_1) = \kappa_{\mathfrak{C}(\mathbf{K})}(\omega_3) = \{s_1\}$,

$\kappa_{\mathfrak{C}(\mathbf{K})}(\omega_4) = \kappa_{\mathfrak{C}(\mathbf{K})}(\omega_5) = \text{Max}\ \{\{s_2\}, \{s_2, s_4\}\} = \{s_2, s_4\}$

$\kappa_{\mathfrak{C}(\mathbf{K})}(\omega_6) = \kappa_{\mathfrak{C}(\mathbf{K})}(\omega_7)$
$= \text{Max}\ \{\{s_3\}, \{s_4\}, \{s_3, s_4\}\} = \{s_3, s_4\}$ ■

Example 2

Let us assume that the set of propositional symbols in the language is $\{p_1, \ldots, p_n\}$. Let $\mathbf{K} = S_1 \cup \ldots \cup S_n$ be a stratified knowledge base where $S_i = \{\neg p_i, p_i\}$.

The labelled knowledge base associated to **K** is:

$$\mathbf{K'} = \cup_{i=1,n}\{(p_i : \{s_{i1}\}), (\neg p_i : \{s_{i2}\}),$$

Let $\leq_{\mathcal{G}}$ the proposed ordering associated to **K'** in order to recover the lexicographical system.

It is easy to check that:

$$\mathfrak{C}(\mathbf{K'}) = \{(\bot : \cup_{i=1,n}\{s_{i1}\}\}$$

This can be shown by noticing that combining two beliefs of each level together leads to have:

$$\{(p_i : \{s_{i1}\})\} \Diamond \{(\neg p_i : \{s_{i2}\})\}$$
$$= \{(p_i : \{s_{i1}\}\} \cup \{(\neg p_i : \{s_{i2}\}) \cup \{(\top : \{s_{i1}, s_{i2}\})$$
$$\equiv \{(\bot : \{s_{i1}\}\} \text{ (using Prop. 9)}$$

and noticing that for any $i \neq j$, we have:
$\{(p_i:\{s_{i1}\}\} \lozenge \{(\neg p_i:\{s_{i2}\})\} \lozenge \{(p_j:\{s_{j1}\})\} \lozenge \{(\neg p_j:\{s_{j2}\})\}$
$=\{(\bot: \{s_{i1}\}\} \cup \{(\bot: \{s_{j1}\}\} \cup \{(\bot: \{s_{i1}, s_{j1}\}\}$
$\equiv \{(\bot: \{s_{i1}, s_{j1}\}\}$ ■

4.2. Syntactic entailment

Until now, we have achieved the first step of our approach to inconsistency handling, namely, the transformation of the original knowledge base **K** by syntactic combination into $\mathbb{C}$(**K**). It has resulted in introducing new formulas with "intermediary" levels. In this section, we propose to applying a syntactic inference $\vdash_\kappa$ which is complete and sound with respect to the κ–semantics, namely:

$\mathbb{C}(\mathbf{K})\cup\{(\alpha : \mathcal{S})\} \vdash_\kappa \beta$ iff $\alpha \models_{\kappa,\mathbb{C}(\mathbf{K})} \beta$.

Let $\mathbb{L}_{\mathbb{C}(\mathbf{K})} = \{I \mid (\phi: I)\in \mathbb{C}(\mathbf{K}) \cup \{(\alpha : \mathcal{S})\}\}$ be the set of labels appearing in $\mathbb{C}(\mathbf{K}) \cup \{(\alpha : \mathcal{S})\}$. To define the syntactic inference $\vdash_\kappa$, we will distinguish two cases, depending if the preference relation $\leq_\mathcal{S}$ defined in $\mathbb{L}_{\mathbb{C}(\mathbf{K})}$ is complete (i.e., $\forall I,J\subseteq\mathbb{L}_{\mathbb{C}(\mathbf{K})}$, we have either $I \leq_\mathcal{S} J$ or $J \leq_\mathcal{S} I$), or not (i.e., it may exist incomparable elements in $\mathbb{L}_{\mathbb{C}(\mathbf{K})}$).

4.2.1. Case of a complete pre-ordering

When $\leq_\mathcal{S}$ defined in $\mathbb{L}_{\mathbb{C}(\mathbf{K})}$ is complete then the entailment $\vdash_\kappa$ is efficient. In fact, when $\leq_\mathcal{S}$ is complete, the entailment $\vdash_\kappa$ coincides with the possibilistic logic entailment (Dubois et al., 1994). The entailment $\vdash_\kappa$ is performed by means of a labelled version of the resolution principle, a counterpart of the possibilistic resolution principle (Dubois et al., 1994):

$$\frac{\begin{array}{ll}(\beta \vee \gamma: & I)\\ (\neg\beta \vee \delta: & J)\end{array}}{(\gamma \vee \delta:\ \mathrm{Min}(I, J))}$$

When $I=_\mathcal{S}J$ then Min(I, J) is arbitrary either I or J (this has no incidence on the inference), as it is justified by the following proposition:

Proposition 12: Let $(\phi_1 : I_1) \in \mathbb{C}(\mathbf{K})$ and $I_1=_\mathcal{S}I_2$. Let **K'** be a new labelled knowledge base constructed from $\mathbb{C}$(**K**) by replacing the formula $(\phi_1 : I_1)$ by the labelled formula $(\phi_1 : I_2)$. Then **K'** and $\mathbb{C}$(**K**) induce the same κ-ranking on **Ω**.

Building the entailment on the resolution principle suggests that only formulas under the clausal form can be used. The following proposition shows that κ-semantics is not sensitive to putting labelled formulas under their clausal form:

Proposition 13: Let $(\phi : I)\in \mathbb{C}(\mathbf{K})$. Let $\{c_1,\ldots,c_n\}$ be the clausal form of ϕ (i.e., all c_1's are clauses and that $\phi\equiv c_1\wedge\ldots\wedge c_n$). Let **K'** be the labelled knowledge base obtained from $\mathbb{C}$(**K**) by replacing $(\phi : I)$ by the set of labelled formulas $\{(c_1 : I),\ldots,(c_n : I)\}$. Then $\mathbb{C}$(**K**) and **K'** are equivalent, i.e., they induce the same κ-ranking on Ω, namely $\forall\omega\in\Omega$, $\kappa_{\mathbb{C}(\mathbf{K})}(\omega) = \kappa_{\mathbf{K'}}(\omega)$.

To check if a given formula ϕ can be considered as a plausible consequence of $\mathbb{C}$(**K**) and a completely certain information α, denoted by $\mathbb{C}(\mathbf{K})\cup\{(\alpha : \mathcal{S})\}\vdash_\kappa \phi$, we proceed by refutation as in classical logical logic:

- First we compute $\mathrm{Inc}(\mathbb{C}(\mathbf{K})\cup\{(\alpha : \mathcal{S})\}) = \mathrm{Max}\{I \mid \mathbb{C}(\mathbf{K})\cup\{(\alpha : \mathcal{S})\} \vdash_\kappa (\bot: I)\}$ the inconsistency level of $\mathbb{C}(\mathbf{K})\cup\{(\alpha : \mathcal{S})\}$ which is the greatest label associated to the empty clause obtained by successive applications of the above resolution principle;
- Next, we again compute $\mathrm{Inc}(\mathbb{C}(\mathbf{K})\cup\{(\alpha : \mathcal{S})\cup\{(\neg\phi : \mathcal{S})\})$ when we add to $\mathbb{C}(\mathbf{K})\cup\{(\alpha : \mathcal{S})\}$ the assumption that ϕ is completely false. Then we have:

$\mathbb{C}(\mathbf{K})\cup\{(\alpha : \mathcal{S})\}\vdash_\kappa \phi$ iff
$\mathrm{Inc}(\mathbb{C}(\mathbf{K})\cup\{(\alpha:\mathcal{S})\}) <_\mathcal{S} \mathrm{Inc}(\mathbb{C}(\mathbf{K})\cup\{(\alpha:\mathcal{S})\cup\{(\neg\phi:\mathcal{S})\})$.

In possibilistic logic, this inference method is as efficient as classical logic refutation by resolution, and can be implemented in the form of an A*-like algorithm (Dubois et al., 1987). The inference needs at most $\log_2(n)$ satisfiability checks where n is the number of equivalent classes in $\mathbb{L}_{\mathbb{C}(\mathbf{K})}$ induced by $\leq_\mathcal{S}$ ("n" in the worst case is equal to $|\mathbb{L}_{\mathbb{C}(\mathbf{K})}|$).

We can easily check that the syntactic entailment $\vdash_\kappa$ can be encoded in classical logic via a thresholding step:

$\mathbb{C}(\mathbf{K})\cup\{(\alpha : \mathcal{S})\}\vdash_\kappa \phi$ iff $\mathbf{BC} \vdash \phi$,

where **BC** is:

$\mathbf{BC} = \{\psi \mid (\psi : I) \in \mathbf{K}\cup\{(\alpha : \mathcal{S})\}$ and $\mathrm{Inc}(\mathbf{K}\cup\{(\alpha : \mathcal{S})\}) <_\mathcal{S} I\}$.

This means, for instance, that once the resulting knowledge base $\mathbb{C}$(**K**) from a stratified inconsistent **K** has been computed, this way of handling inconsistencies leads to a coherence approach where only one consistent subbase of $\mathbb{C}$(**K**) is selected to perform inference, applying possibilistic logic when $\leq_\mathcal{S}$ is a complete pre-ordering. More important is the fact that this inference method is sound and complete with respect to κ-semantics (Dubois et al., 1987). Namely :

$\alpha \models_{\mathbf{L},\mathbf{K}} \beta$ iff $\alpha \models_{\kappa,\mathbb{C}(\mathbf{K})} \beta$ iff $\mathbb{C}(\mathbf{K})\cup\{(\alpha : \mathcal{S})\}\vdash_\kappa \phi$.

Hence our approach supplies a theorem proving technique for lexicographic inconsistent-tolerant inference and penalty logic, whose efficiency is similar to classical logic. It also supplies a theorem prover for an infinitesimal version of Shafer's [Shafer, 1976] evidence theory, since the latter is equivalent to penalty logic. This equivalence can be shown as follows: let $\mathbf{K}=\{(\phi_i\ c_i)/\ i\in I\}$ be a penalty knowledge base. We take an arbitrary infinitesimal δ and let, for each $(\phi_i\ \ c_i)\in \mathbf{K}$, we associate a simple support function m_i defined by: $m_i(\phi_i)=1-\delta^{c_i}$, $m_i(\Omega)=\delta^{c_i}$, $m_i(\text{elsewhere})=0$. From these m_i's we build a combined mass assignment $m_\oplus$ by Dempster's rule, i.e, $m_\oplus = \oplus\{m_i \mid i\in I\}$. Then we can check that $\alpha\vdash_{\text{pen},\mathbf{K}} \beta$ iff $\alpha\vdash_{\mathbf{K},\mathbb{C}(\mathbf{K}_{\text{pen}})} \beta$ iff $\lim_{\delta\to 0}\text{Bel}(\beta|\alpha)=1$. See [Benferhat et al., 1995b], [Dupin et al., 1994] for more details.

Example 1 (continued)
Let $\mathbf{K}=\{(a : \{s_1\}), (\neg a\vee b\text{: } \{s_2\}), (\neg a\vee\neg b\text{: } \{s_3\}), (\neg a\wedge c\text{: } \{s_4\})\}$, and $\mathfrak{C}=\{\{s_2,s_3\}<\{s_1\}, \{s_2\}\le\{s_3\}, \{s_3\}\le\{s_2\}, \{s_1\}<\{s_4\}\}$. We recall that:

$\mathbb{C}(\mathbf{K}) = \{(a : \{s1\}),\ (a\vee c : \{s1, s_4\})\},$
$\quad(\neg a\vee b : \{s_2, s_4\}), (\neg a\vee\neg b : \{s_3, s_4\})\}$

We are looking for the plausible consequences of $\mathbb{C}(\mathbf{K})$ and $(\alpha=\neg b\wedge\neg c\text{: } \mathcal{S})$. We first add the two clauses $(\neg b\text{: } \mathcal{S})$ and $(\neg c\text{: } \mathcal{S})$ to $\mathbb{C}(\mathbf{K})$.
We have:

$\mathbb{L}_{\mathbb{C}(\mathbf{K})} = \{\mathcal{S}, \{s1\}, \{s1, s_4\}, \{s_2, s_4\}, \{s_3, s_4\}\}.$

Note that $\le_{\mathcal{S}}$ defined on $\mathbb{L}_{\mathbb{C}(\mathbf{K})})$ is complete, as it is illustrated by the following figure:

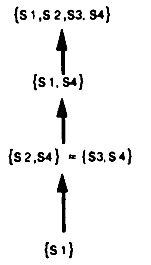


Figure 5

Let us compute the consistent subbase **BC** by first computing $\text{Inc}(\mathbb{C}(\mathbf{K})\cup\{(\alpha : \mathcal{S})\})$. The following derivation gives the optimal label associated to $\perp$:

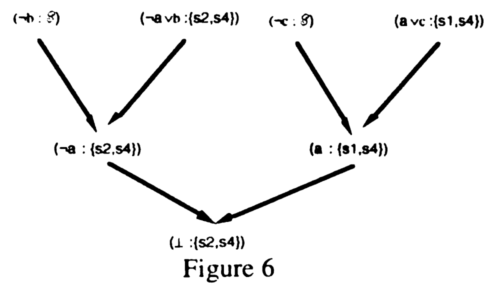


Figure 6

We can check that $\{s_2, s_4\}$ is the highest rank associated to the empty set. Finally, the consistent subbase is:

$\mathbf{BC} = \{\neg b\wedge\neg c,\ a\vee c\},$

and we can easily check that:

$\forall\beta,\ \mathbb{C}(\mathbf{K}) \cup (\neg b\wedge\neg c\text{: } \mathcal{S}) \vdash_{\mathbf{K}} \beta$ iff $\mathbf{BC} \vdash \beta$. ■

4.2.2. Case of a non-complete pre-ordering

When the preference relation $\le_{\mathcal{S}}$ is not complete then computing $\text{Inc}(\mathbb{C}(\mathbf{K})\cup\{(\alpha : \mathcal{S})\})$ is more tricky. In this section, we briefly describe how the previous machinery can be extended, and the complete description of this machinery in the case of partial ordering is left for further research. First, $\text{Inc}(\mathbb{C}(\mathbf{K})\cup\{(\alpha : \mathcal{S})\})$ is generally a set of labels and not only a label.
To extend the previous machinery to the case of partial ordering, we first transform each belief $(\phi : I)$ into a formula $(\phi\text{: } \{I\})$. Namely, in this section a label is not a subset of $\mathcal{S}$, but a family of subsets of $\mathcal{S}$. Indeed, when performing resolution, Min(I, J) is in general a family of subsets of $\mathcal{S}$. We will denote $\mathscr{A}, \mathscr{B}, \ldots$ as families of subsets of $\mathcal{S}$. The resolution principle given above is extended simply in the following way:

$$\frac{\begin{array}{ll}(\beta\vee\gamma\text{:} & \mathscr{A})\\ (\neg\beta\vee\delta\text{:} & \mathscr{B})\end{array}}{(\gamma\vee\delta\text{: } \text{Min}(\mathscr{A}\cup\mathscr{B}))}$$

We call an inconsistent environment a family of subsets of $\mathcal{S}$ associated to the empty clause and obtained from $\mathbb{C}(\mathbf{K})\cup\{(\alpha : \{\mathcal{S}\})\}$ using the above resolution principle.
Then, we define **BC** as follows: a belief ϕ is in **BC** if $(\phi\text{: } \{I\})\in \mathbb{C}(\mathbf{K})\cup\{(\alpha : \{\mathcal{S}\})\}$ and for each inconsistent environment $\mathscr{A}$, and for each $J\in\text{Min}(\mathscr{A})$, $I >_{\mathcal{S}} J$.

Example:
Let $\mathbf{K}=\{(a : \{s_1\}), (b\text{: } \{s_2\}), (\neg a\vee\neg b\text{: } \{s_3\})\}$, and $\mathfrak{C}=\varnothing$. The closure of $\mathfrak{C}$ is simply the one induced by the axioms A1-A3. Note that the inference $\vdash_{\mathbf{L},\mathbf{K}}$ recovers in fact Rescher and Manor's (1970) universal inference, or equivalently $\vdash_{\text{Incl},\mathbf{K}}$ where there is no stratification.
Let us compute $\mathbb{C}(\mathbf{K})$. We have:

$\{(a\text{:}\{s_1\})\} \lozenge \{(b\text{:}\{s_2\})\} = \{(a\text{:}\{s_1\}),(b\text{:}\{s_2\}),(a\vee b\text{:}\{s_1, s_2\})\}$

$\{(a\text{:}\{s_1\})\} \lozenge \{(b\text{:}\{s_2\})\} \lozenge \{(\neg a\vee\neg b\text{: } \{s_3\})\}$
$= \{(a\text{:}\{s_1\}),(b\text{:}\{s_2\}),(a\vee b\text{:}\{s_1, s_2\}), (\neg a\vee\neg b\text{: } \{s_3\}),$
$\quad(\top\text{: } \{s_1, s_3\}), (\top\text{: } \{s_2,s_3\}), (\top\text{: } \{s_1, s_2,s_3\})\}$
$\equiv\{(a\text{:}\{s_1\}),(b\text{:}\{s_2\}),(a\vee b\text{:}\{s_1, s_2\}), (\neg a\vee\neg b\text{: } \{s_3\})\}.$

We have:

$\mathbb{L}_{\mathbb{C}(\mathbf{K})}) = \{\mathcal{S}, \{s_1\}, \{s_2\},\ \{s_3\}, \{s_1, s_2\}\}.$

Note that $\leq_{\mathcal{G}}$ defined on $\mathbb{L}_{\mathbb{C}(\mathbf{K})})$ is not complete as it is explained by the following figure:

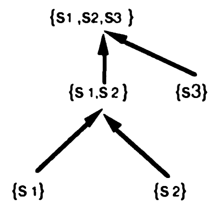


Figure 7

Let us check what are the plausible consequences of $\mathbb{C}(\mathbf{K})$ (we assume that we have no additional information, i.e., $\alpha=\top$). We can easily check that in $\mathbb{C}(\mathbf{K})$ we only have one inconsistent environment with label $\mathcal{A}$:

$\mathcal{A}= \{\{s_1\}, \{s_2\}, \{s_3\}\}$.

Therefore:

BC = $\{a \vee c\}$. ■

5. CONCLUDING DISCUSSIONS

Several classical, prioritized methods for inconsistency-tolerant reasoning, based on the selection of one or several consistent subbases have been unified in the framework of a label calculus based on a general partial preordering between sets satisfying prescribed properties of compatibility with inclusion and cancellation. Thus, the approach takes place among the labelled deductive systems (Gabbay, 1991). The originality of this approach lies in the fact that different kinds of preference relations on subbases (among which inclusion-based and cardinality-based ones) are viewed as special case of a more general preordering of subbases, defined by means of constraints extended by rationality properties. It is noticeable that the cancellation property $I \cup J <_{\mathcal{G}} I \cup K \Leftrightarrow J <_{\mathcal{G}} K$ for pairwise disjoint sets is the one of comparative probabilities [Fishburn, 1986], although we remain in the symbolic framework. Moreover such kind of orderings also appear as natural subjective comparative probabilities in qualitative decision theory [Dubois et al., 1997]. We have shown how to turn a prioritized, or penalty-labelled knowledge base into a semantically equivalent one where inconsistency-tolerant inference can be obtained using resolution-based refutation in the style of possibilistic logic. This equivalent knowledge-base is obtained by a symmetric and associative syntactic combination that can process one formula after the other in an incremental fashion.

Reasoning and acting in the face of inconsistent information is a pressing need for applications involving agents (both software or physical) which must operate in dynamic and partly unknown environments. These systems must be able to cope with the occurrence of unexpected contingencies. Consider for instance a robot operating in an unmodified real-world environment. Due to the inevitable limitations of the prior information and of the sensing apparatus, the robot must rely on uncertain and tentative information about the environment in order to plan its actions. This information may later be contradicted by the new data acquired by the sensors; for example, a corridor may be blocked by an obstacle, and the robot must try an alternative route. The usual approach to cope with unexpected contingencies is to generate plans which explicitly account for the possibility that this or that assumptions turn out to be wrong, for instance, a plan may incorporate a monitor to cheçk that corridors are not blocked. This approach has the drawback that all possible causes for failure must be known in advance. Non-monotonic systems based on prioritized knowledge-bases may offer an alternative way to cope with these situations: the agent is prepared to the possibility of having an inconsistent set of beliefs, and can reason from it to derive an appropriate reaction, without having to explicitly list each cause of inconsistency and each repairing action. (In a sense, this list is replaced by a preference order between the defaults used in the reasoning process.) Prioritized systems, however, tend to be computationally expensive: the compilation technique presented in this work may help in reducing complexity when the queries to the knowledge base are more frequent than the modifications done -- which is usually the case for the coarse-grained environment models used by robot's planners.

Although our proposal simplifies the problem of computing inconsistency tolerant inference by avoiding an explicit enumeration of maximal consistent subsets, some difficulties remain because the intrinsic complexity of inconsistency-tolerant inference has been shifted elsewhere. This technique is actually a kind of "knowledge compilation" (similarly to the computation of prime implicants of a knowledge base, or ATMS labels) where an increase of spatial complexity is compensated by a gain in temporal complexity especially when the inconsistent knowledge base is used repeatedly with different queries. Moreover, the inference procedure may require as a subproblem the computation of minimal or maximal elements in a poset, each time a resolution step is taken. The case is simpler with penalty logic or lexicographic inference that reduce to a plain possibilistic logic-like machinery. Lastly, the process of putting the labelled knowledge base in some standard format may be computationally expensive. A future work will consist in defining methods to reduce the size of $\mathbb{C}(\mathbf{K})$ which can be large in the worst case (even if subsumption is used and redundancies are eliminated). Of course $\mathbb{C}(\mathbf{K})$ can be computed once for all, and the incrementality of combination efficiently copes with the dynamics of the knowledge base.

Another future work is to see if our approach can cover argument-based reasoning techniques such as the ones of Benferhat et al. [1993; 1995a], Fox et al. [1992] or Amgoud et al. [1996], Cayrol [1995] which are not coherence-based in the sense of this paper.

References

L. Amgoud, C. Cayrol, D. Le Berre (1996) Comparing arguments using preference orderings for argument-based reasoning. Proc. IEEE International Conf. on Tools in AI (ICTAI'96), Toulouse, pp. 400-403

S. Benferhat, C. Cayrol, D. Dubois, J. Lang, H. Prade (1993a) Inconsistency management and prioritized syntax-based entailment. Proc. of the 13th Inter. Joint Conf. on Artificial Intelligence (IJCAI'93), pp. 640-645.

S. Benferhat, D. Dubois, H. Prade (1993b) "Argumentative inference in uncertain and inconsistent knowledge bases" Proc. of the 9th Conf. on Uncertainty in Artificial Intelligence, Washington, Morgan & Kaufmann, San Mateo, CA, pp. 411-419,

S. Benferhat , D. Dubois, H. Prade (1995a) How to infer from inconsistent beliefs without revising?. Proc. of the 14th Inter. Joint Conf. on Artificial Intelligence (IJCAI'95), Montréal, Canada, 1449-1455.

S. Benferhat, A. Saffiotti and P. Smets (1995b). Belief functions and default reasoning. Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence, Montreal, CA, pp. 19-26.

S. Benferhat, D. Dubois et H. Prade (1997) From semantic to syntactic approaches to information combination in possibilistic logic". In Aggregation and Fusion of Imperfect Information, Studies in Fuzziness and Soft Computing (B. Bouchon-Meunier, ed.), Physica Verlag, pp. 141-151.

Boldrin L., Sossai C. (1995) An algebraic semantics for possibilistic logic. Proc of the 11th Conf. Uncertainty in Artifucial Intelligence (P. Besnard, S. Hank, eds.) Morgan Kaufmann, San Francisco, CA, 27-35.

G. Brewka (1989) Preferred subtheories: an extended logical framework for default reasoning. Proc. of the 11th Inter. Joint Conf. on Artificial Intelligence (IJCAI'89), Detroit, 1043-1048.

C. Cayrol (1995) On the relation between argumentation and nonmonotonic coherence based entailment. Proc. of the 14th Inter. Joint Conf. on Artificial Intelligence (IJCAI'95) Montreal Canada, 1443-1448.

J. De Kleer, (1990) Using crude probability estimates to guide diagnosis. Artificial Intelligence, 45, 381-391.

D. Dubois, H. Fargier, H. Prade (1997) Decision-making under ordinal preferences and uncertainty. Revised version in: Proc. of the 13th Conf. on Uncertainty in Artificial Intelligence (D. Geiger, P.P. Shenoy, eds.), Providence, RI,, Morgan & Kaufmann, San Francisco, CA, 157-164.

D. Dubois, J. Lang, H. Prade (1987) Theorem-proving uncer uncertainty – a possibility theory-based approach. Proc. of the 10th Inter. Joint Conf. on Artificial Intelligence (IJCAI'87), Milano, 984-986.

D. Dubois, J. Lang, H. Prade (1994) Possibilistic logic. In: Handbook of Logic in Artificial Intelligence and Logic Programming, Vol. 3 (D.M. Gabbay et al., eds.), Oxford University Press, 439-513.

D. Dubois, H. Prade (1987) Necessity measures and the resolution principle. IEEE Trans. on Systems, Man and Cybernetics, 17(1), 474-476.

F. Dupin de St Cyr , J. Lang , T. Schiex (1994) Penalty logic and its link with Dempster-Shafer theory. Proc. of the 10th Conf. on Uncertainty in Artificial Intelligence, Morgan & Kaufmann, 204-211

R. Fagin J.D. Ullman, M.Y. Vardi (1983) On the semantics of updates in databases. Proc. 2d ACM SIGACT-SIGMOD Symp. on the proinciples of Databases Systems, Atlanta, 352-365.

P. Fishburn (1986) The axioms of subjective probabilities. statistical science 1, 335-358.

J. Fox, P. Krause, S. Ambler (1992) Arguments, contradictions and practical reasoning. Proc. of the 10th European Conf. on Artificial Intelligence (ECAI'92), Vienna, Austria, 623-627.

D. M. Gabbay (1991) Labelled Deductive Systems. Oxford University Press, 1991.

Goldszmidt, M. and Pearl, J. (1991) System Z^+: A formalism for reasoning with variable-strength defaults. Procs. of the AAAI Conference, Anaheim, CA, pp. 399-404.

D. Lehmann (1995) Another perspective on default reasoning. Annals of Mathematics and Artificial Intelligence, 15, pp. 61-82.

D. Lehmann (1996) Generalized qualitative probability: Savage revisited. In Procs. of 12th Conference on Uncertainty in Artificial Intelligence (UAI-96), (E. Horvitz and F. Jensen eds.), pp. 381-388.

B. Nebel(1991), Belief revision and default reasoning : syntax-based approaches. Proceedings of the 2nd Inter. Conf. on Principles of Knowledge Representation and Reasoning KR'91 (J. Allen, R. Fikes, E Sandewall, eds.), Morgan-Kaufmann, Cambridge, MA, 417-428.

B. Nebel (1994) Base revision operator ans schemes: semantics representation and complexity. Proc. of the 11th European Conference on Artificial Intelligence, Amsterdam, pp.341-345.

G. Pinkas, R.P. Loui (1992) Reasoning from inconsistency: a taxonomy of principles for resolving conflict. Proc. of KR'92 Cambridge, Mass., 709-719.

N. Rescher (1976) Plausible reasoning. Van Gorcum, Amsterdam.

N. Rescher, R. Manor (1970) On inference from inconsistent premises. Theory and Decision, 1, 179-219.

G. Shafer (1976) A Mathematical Theory of Evidence. Princeton University Press, Princeton, NJ.

Y. Shoham (1988) Reasoning about Change. The MIT Press, Cambridge, MA.

W. Spohn, Ordinal conditionalfunctions: A dynamic theory of epistemic states. In: Causation in Decision,Belief Change, and Statistics (W.L.Harper, B.Skyrms, eds.), Kluwer Acad.Pub. 105-134.

M.A. Williams (1996). Toward a practical approach to belief revision: reason-based change. Proc. of the 4th Inter. Conf. on Principles of Knowledge Representation and Reasoning (KR'96).

Appendix : (Proofs of technical propositions)

Proposition 2: $\alpha \models_{lex,\mathbf{K}} \beta$ iff $\alpha \models_{L,\mathbf{K}_{Lex}} \beta$.

Proof:

Let $\mathbf{K}_{Lex} = \{(\phi_{ij} : \{s_{ij}\})\}$ be the labelled knowledge base associated to a stratified knowledge base $\mathbf{K}=S_1\cup\ldots\cup S_n$. Let $\leq_{\mathcal{S}}$ the proposed ordering associated to $\mathbf{K}_{Lex}$ in order to recover the lexicographic system. Let us show that: $\alpha \models_{lex,\mathbf{K}} \beta$ iff $\alpha \models_{L,\mathbf{K}_{Lex}} \beta$.

It is enough to show that : $L_{\mathbf{K}_{Lex}}(\omega) < L_{\mathbf{K}_{Lex}}(\omega')$ iff $f(\omega)$ is lex-preferred to $f(\omega')$ (and for simplicity we say ω is lex-preferred to ω'), where: $f(\omega)$ is the set of formulas in K satisfied by ω. We recall that:

$$L_{\mathbf{K}_{Lex}}(\omega)=\cup_{k=1,n} A(\omega;k)$$

where $A(\omega; k)$ is the set of labels of the formulas in S_k falsified by ω. We can easily state that ω is lex-preferred to ω' iff $\forall j>i$; $|A(\omega; j)|=|A(\omega'; j)|$; and $|A(\omega; i)|<|A(\omega'; i)|$.

Consider the "if" part of our thesis. Assume that ω is lex-preferred to ω'. Then;

$$\begin{aligned}L_{\mathbf{K}_{Lex}}(\omega) &= \cup_{k=1,n} A(\omega;k)\\ &= [\cup_{k=1,i-1} A(\omega;k)] \cup A(\omega;i) \cup [\cup_{k=i+1,n} A(\omega;k)]\\ &=_{\mathcal{S}} [\cup_{k=1,i-1} A(\omega;k)] \cup A(\omega;i) \cup [\cup_{k=i+1,n} A(\omega';k)]\\ &\qquad \text{(since } A(\omega';k)=_{\mathcal{S}}A(\omega;k) \text{ for } k>i)\\ &<_{\mathcal{S}} [A(\omega';i) - A(\omega;i)] \cup A(\omega;i) \cup [\cup_{k=i+1,n} A(\omega';k)]\end{aligned}$$

Indeed, from the constraints on labels in the lexicographical case we have:

$$\cup_{k=1,i-1} \mathcal{S}_k <_{\mathcal{S}} \{s_{ij}\}$$

$$\Rightarrow \cup_{k=1,i-1} A(\omega;i) <_{\mathcal{S}} \{s_{ij}\}$$

[using A2 noticing that $\cup_{k=1,i-1} \mathcal{S}_k \supseteq \cup_{k=1,i-1} A(\omega;i)$]

$$\Rightarrow \cup_{k=1,i-1} A(\omega;i) <_{\mathcal{S}} [A(\omega';i)-A(\omega;i)]$$

(since $A(\omega';i)-A(\omega;i)$ contains at least one element of level i; if more than one element we use A2)

Now using the facts that $|A(\omega; i)| < |A(\omega'; i)|$ and that $\{s_{ij}\}=_{\mathcal{S}}\{s_{ik}\}$ for $k\neq j$, we can easily check that:

$$A(\omega';i) =_{\mathcal{S}} A(\omega;i) \cup X$$

such that $X\subset A(\omega';i)$ and $X\cap A(\omega;i)=\emptyset$. Indeed, this can be obtained in the following way: for any element s_{ip} which is in $A(\omega; i)$ but not in $A(\omega'; i)$, we take an element s_{ir} of $A(\omega'; i)$ but not in $A(\omega; i)$, and replace this element s_{ir} by s_{ip} in $A(\omega'; i)$.

Hence:

$L_{\mathbf{K}_{Lex}}(\omega)<_{\mathcal{S}}[(A(\omega;i) \cup X)-A(\omega;i)]\cup A(\omega;i)\cup[\cup_{k=i+1,n} A(\omega';k)]$

$\Rightarrow L_{\mathbf{K}_{Lex}}(\omega) <_{\mathcal{S}} A(\omega';i) \cup [\cup_{k=i+1,n} A(\omega';k)]$

$\Rightarrow L_{\mathbf{K}_{Lex}}(\omega) <_{\mathcal{S}} [\cup_{k=1,i-1} A(\omega';k)] \cup A(\omega';i) \cup [\cup_{k=i+1,n} A(\omega';k)]$ (using A2 and A1)

$\Rightarrow L_{\mathbf{K}_{Lex}}(\omega) <_{\mathcal{S}} L_{\mathbf{K}_{Lex}}(\omega')$

For proving the converse, assume that $L_{\mathbf{K}_{Lex}}(\omega) <_{\mathcal{S}} L_{\mathbf{K}_{Lex}}(\omega')$ and suppose that ω is not lex-preferred to ω'. We distinguish two cases:

• ω' is not lex-preferred to ω either; and hence ω and ω' falsifies exactly the same number of rules in each S_i; but this means that $L_{\mathbf{K}_{Lex}}(\omega)=_{\mathcal{S}} L_{\mathbf{K}_{Lex}}(\omega')$; which contradicts our hypothesis; or

• ω' is lex-preferred to ω; but then; using if part of the proof; $L_{\mathbf{K}_{Lex}}(\omega') <_{\mathcal{S}} L_{\mathbf{K}_{Lex}}(\omega)$; which again contradicts the hypothesis. Hence ω must be lex-preferred to ω'. ■

Proposition 6: If $\mathbb{C}(\mathbf{K}) = \mathbf{K}' \diamond \mathbf{K}''$ then the equivalence $\alpha \models_{L,\mathbf{K}} \beta$ iff $\alpha \models_{\kappa,\mathbb{C}(\mathbf{K})} \beta$ is satisfied.

Proof:

We need to show that: $\forall \omega\in\Omega$, $L_{\mathbf{K}}(\omega)=\kappa_{\mathbf{K}'\diamond\mathbf{K}''}(\omega)$. Consider the four possible cases:

• if ω is a model of $\mathbf{K}'$ and $\mathbf{K}''$ (hence trivially ω is also a model of $\mathbf{K}'\vee\mathbf{K}''$) then $L_{\mathbf{K}}(\omega)=\emptyset$. This also means that: $\kappa_{\mathbf{K}'}(\omega)=\emptyset$, $\kappa_{\mathbf{K}''}(\omega)=\emptyset$, $\kappa_{\mathbf{K}'\vee\mathbf{K}''}(\omega)=\emptyset$. Therefore, we have: $\kappa_{\mathbf{K}'\diamond\mathbf{K}''}(\omega)=\mathrm{Max}\{\kappa_{\mathbf{K}'}(\omega), \kappa_{\mathbf{K}''}(\omega), \kappa_{\mathbf{K}'\vee\mathbf{K}''}(\omega)\}$
$=\emptyset=L_{\mathbf{K}}(\omega)$.

• if ω is a model of $\mathbf{K}'$ (hence trivially ω is also a model of $\mathbf{K}'\vee\mathbf{K}''$) but not of $\mathbf{K}''$ then $L_{\mathbf{K}}(\omega)=L_{\cup_{i=j+1,n}\mathbf{K}_i}(\omega)$. Moreover, in this case: $\kappa_{\mathbf{K}'}(\omega)=\emptyset$, $\kappa_{\mathbf{K}'\vee\mathbf{K}''}(\omega)=\emptyset$. Hence we have:

$$\begin{aligned}\kappa_{\mathbf{K}'\diamond\mathbf{K}''}(\omega) &= \mathrm{Max}\{\kappa_{\mathbf{K}'}(\omega), \kappa_{\mathbf{K}''}(\omega), \kappa_{\mathbf{K}'\vee\mathbf{K}''}(\omega)\}\\ &= \kappa_{\mathbf{K}''}(\omega)\\ &= L_{\cup_{i=j+1,n}\mathbf{K}_i}(\omega) \text{ (by assumption)} = L_{\mathbf{K}}(\omega).\end{aligned}$$

• if ω is a model of $\mathbf{K}''$ but not of $\mathbf{K}'$ then the thesis holds as in the previous case.

• The last case is when ω is neither a model of $\mathbf{K}'$ nor of $\mathbf{K}''$, then let :

$$M=\mathrm{Max}\{I_k / (\phi_k: I_k)\in \mathbf{K}', \omega\not\models \phi_k\}, \text{ and}$$
$$N =\mathrm{Max}\{I_m/ (\phi_m: I_m)\in \mathbf{K}'', \omega\not\models \phi_m\}.$$

Note that M and N are unique. we have :

$\kappa_{\mathbf{K}'\diamond\mathbf{K}''}(\omega) = \mathrm{Max}\{M, N, \mathrm{Max}\{I_k\cup I_m/(\phi_k\vee\psi_m:I_k\cup I_m)\in \mathbf{K}'\vee\mathbf{K}'', \omega\not\models\phi_k\vee\psi_m\}\}$

due the fact that

$$I_k\cup I_m> I_k \text{ and}$$
$$I_k\cup I_m > I_m,$$

we get:

$$\begin{aligned}\kappa_{\mathbf{K}'\diamond\mathbf{K}''}(\omega) &= \mathrm{Max}\{ I_p/(\phi_p: I_p)\in \mathbf{K}'\diamond\mathbf{K}'' \text{ and } \omega\not\models\phi_p\}\\ &=\mathrm{Max}\{I_k\cup I_m / (\phi_k\vee\psi_m:I_k\cup I_m)\in \mathbf{K}'\vee\mathbf{K}'', \omega\not\models \phi_k\vee\psi_m\}\\ &= M \cup N = L_{\mathbf{K}}(\omega). ■\end{aligned}$$

Formal Theory Building Using Automated Reasoning Tools

Jaap Kamps
Applied Logic Laboratory
Institute for Logic, Language and Computation
University of Amsterdam
Amsterdam, The Netherlands

Abstract

The merits of representing scientific theories in formal logic are well-known. Expressing a scientific theory in formal logic explicates the theory as a whole, and the logic provides formal criteria for evaluating the theory, such as soundness and consistency. On the one hand, these criteria correspond to natural questions to be asked about the theory: is the theory contradiction-free? (is the theory logically consistent?) is the theoretical argumentation valid? (can a theorem be soundly derived from the premises?) and other such questions. On the other hand, testing for these criteria amounts to making many specific proof attempts or model searches: respectively, does the theory have a model? can we find a proof of a particular theorem? As a result, testing for these criteria quickly defies manual processing. Fortunately, automated reasoning provides some valuable tools for this endeavor. This paper discusses the use of first-order logic and existing automated reasoning tools for formal theory building, and illustrates this with a case study of a social science theory, Hage's axiomatic theory of organizations.

1 INTRODUCTION

The theory building methodology outlined in this paper is by no means a new one. The use of formal logic to represent scientific theories dates, at least, back to the logical positivists (Ayer 1959). What is novel in our approach is the extensive use of automated reasoning tools. One of the reasons for the demise of positivism was the inability to put philosophy into practice. The formalization of scientific theories requires a huge amount of tedious calculations that exceed manual processing capabilities. The use of computational tools allows us to transcend these limitations, and bring much of the positivist philosophy to life.

It is part of the research plan at the Applied Logic Laboratory (ALL) to revive formal theory building in the social sciences by using formal logics and by taking advantage of the available computational support. Social scientists usually agree that theories should be logical, but they rarely address the issue, eschewing the difficulties of investigating the logical structure of 'discursive theories' (theories expressed in natural language, the standard representation in the social sciences). Rather than engaging ourselves in abstract, philosophical deliberations,[1] we focus on applied case studies. That is, we 'formalize' actual social science theories by rationally reconstructing them and expressing them in logical form. The resulting formalizations are then tested using logical criteria, a task greatly facilitated by the availability of computational tools.

In Section 2 we will explain the value of formalization in the explication of scientific theories and the role tools from automated reasoning can play in this process. In Section 3 we present a case study that shows how this can be applied in the social sciences, and we end in Section 4 by summarizing our experiences and drawing conclusions.

2 FORMAL THEORY BUILDING

The principal reason for formalizing scientific theories is to clarify and explicate them. Until a scientific theory is expressed in a formal and unambiguous manner,

[1]See for example Adorno et al. (1969) for interesting arguments in favor of, and against the use of formal logic in the social sciences.

it remains open to many interpretations. Provided they can interpret the formalism, a formal exposition of a scientific theory allows readers to understand the theory, to distinguish between alternative readings, to gauge its boundaries, and to compare it with alternatives (Suppes 1968).

We use the classical, axiomatic-deductive notion of a theory. The premises of a theory consist of universal statements (universal laws or empirical generalizations, possibly supplemented with definitions). The theory itself is the deductive closure of the set of premises (Tarski 1956). Theoretical explanations and predictions correspond to deductions from the set of premises (Popper 1959).

We prefer to use this strict notion of a theory over more liberal ones.[2] The reason for this is simple: our main interest is the justification of theories, and we are therefore interested in strict criteria that can be objectively evaluated. This position has clear limitations; many other aspects of theory building require creativity and insight (activities with which logic is rarely associated). Especially the discovery of theories requires different methods than formal logic and deduction. Most of the research in the social sciences is directed at empirical surveys, and the main theme of all methodology textbooks is how to perform such empirical research. Our efforts do not replace, but rather complement this empirical research. As soon as a (tentative version of) a theory is formulated, we have some powerful tools for evaluating it. There are several logical criteria available for evaluating a theory formalized in a logical language, such as the consistency of the theory or soundness of a derivation.

Many of these logical criteria correspond to natural questions which we would like to ask about a scientific theory:

Is the theory contradiction-free? In logical terms, *is the theory logically consistent?* An inconsistent theory has an empty domain: empirically testing an inconsistent theory is futile. If we can construct a model of the premise set then the theory is logically consistent. One may want to include the (spelled-out) theorems, but as long as these are soundly derivable from the premises they cannot make the theory inconsistent.

To establish that a theory is inconsistent, we use a complementary approach: showing that it leads to an absurdity. An inconsistent theory is a trivial theory, in the sense that any statement (*and* its negation) is derivable from it. If we can derive a contradiction from some subset of the premises, the theory is inconsistent.

Is the argumentation of the theory valid? *Can a given theorem be soundly derived from the premises?* If we can prove a given theorem from the premises, the argumentation is sound. That is, if we consider cases in which the premises hold, then the theorem must also hold (i.e, the theorem is a prediction). Conversely, if we consider cases in which the theorem holds, then the premises give an explanation for the theorem (there may be other explanations, although these must satisfy the same soundness criterion).

For proving the fallaciousness of an argument we use a complementary approach: looking for a counterexample. If we can construct a model in which the premises hold but the conjecture does not hold, we have refuted the conjecture. Many scientific theories have, when taken in a literal sense, fallacious argumentation. The severity of such a flaw depends on the following two questions:

(a) What unstated background knowledge is necessary? In any exposition of a theory, a certain amount of background must be 'taken for granted', that is to say, it is assumed to be indisputable as common knowledge. But such information must be explicitly added to a formalization. For example, the author of a theory which describes how properties of organizations vary over time will hardly find it necessary to explicitly state that the *before* relation between time-points is transitive and anti-symmetric, but this axiom may be necessary to formally derive some of the conclusions of the theory. In this case, counterexamples to such a theorem are 'non-intended' models. For example, a counterexample might be a model in which time-points $t_1 < t_2 < t_3 < t_1$ (a model with a circular notion of time, whereas time is conceived as linear). Finding this counterexample will immediate reveal the cause, in this case a missing axiom on the *before* relation.

(b) What assumptions has the author neglected to make? In contrast to the common knowledge of the previous point, occasionally the formalization of a theory reveals a genuine hiatus in the theory. This may have been due to an oversight by the author or perhaps a failure to consider all possible configurations of his/her assumptions. It can also occur that the author expresses him/herself somewhat infelicitously. In this case, counterexamples cannot be as easily dis-

[2]Although this conception of theory is mainly inspired by theories in mathematics and physics, it is applicable to all empirical sciences, including the social sciences (Rudner 1966; Popper 1969).

carded: we have to deal with a genuine counterexample. This may necessitate either strengthening the premises, weakening the conjecture, or even discarding the conjecture altogether. Examining the counterexample(s) provides useful information for deciding between options for refining the theory.

Is the theory falsifiable? *Are the theorems logically contingent?* If no state of affairs can possibly falsify a theory, then it is a waste of time to empirically test the theory. Falsifiability is an essential property of a scientific theory (Popper 1959). If we can construct a model (disregarding all premises) where a theorem of the theory is false, then this theorem is falsifiable. If we can prove a theorem from an empty premise set, the theorem is not falsifiable. A theory that contains at least one falsifiable theorem (and therefore at least one falsifiable premise) is falsifiable.

If we can also construct a model in which the theorem holds (which is always the case for soundly derivable theorems in a consistent theory), then this theorem is satisfiable too. A theorem that is both satisfiable and falsifiable, is contingent: the validity of the theorem is strictly determined by the premises—it is neither a tautology nor a contradiction.

What is the domain that the theory describes? *What do the models of the theory look like?* A model generator can be used to provide candidate models for exploration. This gives insight into the domain which the theory describes.

In practice, testing for logical criteria requires many derivations involving large sets of formulas. In this endeavor, automated reasoning provides some invaluable tools. At ALL we use the following:

Automated theorem provers are programs designed for finding proofs of conjectures. For the case study of this paper, we used OTTER (McCune 1994b), a resolution-style theorem prover for first-order logic with equality. This theorem prover can find inconsistencies in the set of input formulas. Its principal use is to construct refutation proofs of conjectures, by feeding it a (consistent) premise set together with the negation of a conjecture. If the program derives an inconsistency, then we have, in fact, proved the conjecture (because the conjecture must hold in all models of the premises).

Automated model generators are programs that can enumerate the finite (small) models of a theory. For the case study of this paper, we used MACE (McCune 1994a), a model generator for first-order logic with equality, based on a Davis-Putnam procedure for propositional satisfiability testing. This model generator can find small models of the set of input formulas (for example to prove the consistency of the theory). It can also be used to find counterexamples to a conjecture, by feeding it a premise set together with the negation of the conjecture.

There are many other automated theorem provers and model generators available. We choose here to use OTTER and MACE because they are companion programs that can read the same input format. This is a great advantage for our work, since we want to switch between theorem proving and model generation, depending on which type of tool is most suitable for the specific proof/disproof attempt at hand. In the next section, we will explain the use of these tools for formal theory building in detail.

3 A SOCIAL SCIENCE CASE STUDY

Several case studies of formalization using automated reasoning support tools have been performed. These case studies include the following social science theories: Mintzberg's *Contingency Theory* (Glorie et al. 1990), Thompson's *Organizations in Action* (Kamps and Pólos 1998), and Hannan and Freeman's *Organizational Ecology* including their theory of organizational inertia (Péli et al. 1994), life history strategies (Péli and Masuch 1997), niche width (Bruggeman 1997; Péli 1997), and age dependence fragment (Hannan 1997).[3] These and most other social science theories are stated in natural language. As a result, the main obstacle for formalizing such a discursive theory is their rational reconstruction: interpreting the text, singling out important concepts, distinguishing assumptions and theorems from other parts of the text, and reconstructing the argumentation. This motived our choice for the theory we want to formalize as a case study in this paper: Hage (1965) *An Axiomatic Theory of Organizations.* Although this is not a formal theory in the sense that it uses natural language exclusively, it is an axiomatic theory in which axioms and theorems are clearly outlined.[4] This greatly facilitates

[3]Some fragments of the resulting formalizations are included in the TPTP (Thousands of Problems for Theorem Provers) Problem Library (Sutcliffe et al. 1994).

[4]Note that this is an exceptional case; only few social scientists state carefully formulated propositions, and even fewer authors attempt to make their underlying assumptions explicit. It is also possible to reconstruct less explicit

the rational reconstruction of the theory, and allows us to focus on the actual formalization of the theory and the role automated reasoning tools can play in this process.

3.1 HAGE'S AXIOMATIC THEORY OF ORGANIZATIONS

Hage's axiomatic theory of organizations postulates seven axioms based on the theoretical writing of Weber, Barnard, and Thompson and predicts 21 derived theorems. The theory concerns interrelations between eight organizational variables: *complexity*, *centralization*, *formalization*, *stratification*, *adaptiveness*, *production*, *efficiency*, and *job satisfaction*. Hage (1965, p.293) lists two 'indicators' for each of the eight variables:

Complexity Number of occupational specialties. Level of training required.

Centralization Proportion of jobs that participate in decision making. Number of areas in which decisions are made by decision makers.

Formalization Proportion of jobs that are codified. Range of variation allowed within jobs.

Stratification Differences in income and prestige among jobs. Rate of mobility between low- and high-ranking jobs.

Adaptiveness Number of new programs a year. Number of new techniques a year.

Production Number of units produced per year. Rate of increase in units produced per year.

Efficiency Cost per unit of output per year. Amount of idle resources per year.

Job satisfaction Satisfaction with working conditions. Rate of turnover in job occupants per year.

The first four variables are organizational means, and the second four variables are organizational ends.

Hage (1965) postulates seven axioms that interrelate the eight organizational variables. According to Hage, there are 21 theorems derivable from these seven natural language axioms. Table 1 reprints the 7 axioms and 21 corollaries used in (Hage 1965, p.300). We will reconstruct Hage's theory by formalizing it in first-order logic.

theories, especially if the original authors can be consulted. A case in point is (Péli and Masuch 1997).

3.2 A FIRST FORMALIZATION

We represent the eight organizational variables by unary functions, i.e., $\mathsf{comp}(x)$, $\mathsf{cent}(x)$, $\mathsf{form}(x)$, $\mathsf{stra}(x)$, $\mathsf{adap}(x)$, $\mathsf{prod}(x)$, $\mathsf{effi}(x)$, and $\mathsf{jobs}(x)$ respectively. For example, '$\mathsf{cent}(O_1)$' denotes the *centralization* of organization O_1. In this way, we can represent the fact that the *centralization* of O_1 is higher than that of O_2 by '$\mathsf{cent}(O_1) > \mathsf{cent}(O_2)$' (using a strict ordering '$>$'). Now, the fact that ***the higher the centralization, the higher the production*** (A.1) can be represented by:

$$\forall x, y\, [\, \mathsf{cent}(x) < \mathsf{cent}(y) \;\rightarrow\; \mathsf{prod}(x) < \mathsf{prod}(y)\,]$$

Table 2 contains a first-order formalization of the axioms in Table 1. We can verify the consistency of the theory by attempting to generate a model of it. MACE can generate a model of F.1 through F.7 using domain size 2 (see Table 3). It is easy to verify that the assumptions hold in this model: F.1–F.3 hold vacuously, F.4–F.6 hold, and F.7 holds again vacuously. Consequently, theory F.1–F.7 is consistent.[5] In fact, MACE can derive 5398 models on a domain of size 2.

Table 2: The Formal Assumptions.

F.1	$\forall x, y\, [\, \mathsf{cent}(x) < \mathsf{cent}(y) \rightarrow \mathsf{prod}(x) < \mathsf{prod}(y)\,]$
F.2	$\forall x, y\, [\, \mathsf{form}(x) < \mathsf{form}(y) \rightarrow \mathsf{effi}(x) < \mathsf{effi}(y)\,]$
F.3	$\forall x, y\, [\, \mathsf{cent}(x) < \mathsf{cent}(y) \rightarrow \mathsf{form}(x) < \mathsf{form}(y)\,]$
F.4	$\forall x, y\, [\, \mathsf{stra}(x) < \mathsf{stra}(y) \rightarrow \mathsf{jobs}(x) > \mathsf{jobs}(y)\,]$
F.5	$\forall x, y\, [\, \mathsf{stra}(x) < \mathsf{stra}(y) \rightarrow \mathsf{prod}(x) < \mathsf{prod}(y)\,]$
F.6	$\forall x, y\, [\, \mathsf{stra}(x) < \mathsf{stra}(y) \rightarrow \mathsf{adap}(x) > \mathsf{adap}(y)\,]$
F.7	$\forall x, y\, [\, \mathsf{comp}(x) < \mathsf{comp}(y) \rightarrow \mathsf{cent}(x) > \mathsf{cent}(y)\,]$

Table 3: A Model Of Assumptions **F.1** Through **F.7**.

	cent	prod	form	effi	stra	jobs	adap	comp
O_1	0	0	1	0	0	1	1	1
O_2	0	1	1	0	1	0	0	1

But what about theorems? Hage (1965) derives 21 theorems "...by applying the simple rules of the syllogism" (p.299) to the seven axioms. We use OTTER to test whether the theorems can be soundly derived from the axioms F.1 through F.7. For example, C.2

[5]One can argue that theories consisting solely of universally quantified conditional statements are by definition consistent, since one can construct numerous models in which none of the conditions is satisfied, making all axioms vacuously true. Nevertheless, it is comforting that we can construct a less trivial model of the theory.

Table 1: The Assumptions (**A.1**–**A.7**) And Derived Theorems (**C.1**–**C.21**).

A.1	The higher the centralization, the higher the production.
A.2	The higher the formalization, the higher the efficiency.
A.3	The higher the centralization, the higher the formalization.
A.4	The higher the stratification, the lower job satisfaction.
A.5	The higher the stratification, the higher the production.
A.6	The higher the stratification, the lower the adaptiveness.
A.7	The higher the complexity, the lower the centralization.
C.1	The higher the formalization, the higher the production.
C.2	The higher the centralization, the higher the efficiency.
C.3	The lower the job satisfaction, the higher the production.
C.4	The lower the job satisfaction, the lower the adaptiveness.
C.5	The higher the production, the lower the adaptiveness.
C.6	The higher the complexity, the lower the production.
C.7	The higher the complexity, the lower the formalization.
C.8	The higher the production, the higher the efficiency.
C.9	The higher the stratification, the higher the formalization.
C.10	The higher the efficiency, the lower the complexity.
C.11	The higher the centralization, the lower job satisfaction.
C.12	The higher the centralization, the lower the adaptiveness.
C.13	The higher the stratification, the lower the complexity.
C.14	The higher the complexity, the higher job satisfaction.
C.15	The lower the complexity, the lower the adaptiveness.
C.16	The higher the stratification, the higher the efficiency.
C.17	The higher the efficiency, the lower job satisfaction.
C.18	The higher the efficiency, the lower the adaptiveness.
C.19	The higher the centralization, the higher the stratification.
C.20	The higher the formalization, the lower job satisfaction.
C.21	The higher the formalization, the lower the adaptiveness.

states that *the higher the centralization, the higher the efficiency*. A formal version of the second theorem would read:

$$\forall x, y \, [\, \mathsf{cent}(x) < \mathsf{cent}(y) \rightarrow \mathsf{effi}(x) < \mathsf{effi}(y) \,]$$

This theorem, T.2, is derivable from axiom F.3 and F.2 using the input-file:

```
set(auto).
formula_list(usable).
% F.1
all x y (form(x)<form(y) -> effi(x)<effi(y)).
% F.3
all x y (cent(x)<cent(y) -> form(x)<form(y)).
% negation of T.2
-( all x y (cent(x)<cent(y) -> effi(x)<effi(y)) ).
end_of_list.
```

OTTER's resolution-style proofs require the theorem to be negated in the input-file. If now the theorem (in its original form) is indeed derivable, we will have a contradiction (by *reductio ad absurdum*).

Surprisingly, we can only derive a small fraction of the claimed theorems, namely the three theorems T.2, T.6, and T.7 in Table 4.[6]

Table 4: Theorems Derivable From **F.1** Through **F.7**.

T.2	$\forall x, y \, [\, \mathsf{cent}(x) < \mathsf{cent}(y) \rightarrow \mathsf{effi}(x) < \mathsf{effi}(y) \,]$
T.6	$\forall x, y \, [\, \mathsf{comp}(x) < \mathsf{comp}(y) \rightarrow \mathsf{prod}(x) > \mathsf{prod}(y) \,]$
T.7	$\forall x, y \, [\, \mathsf{comp}(x) < \mathsf{comp}(y) \rightarrow \mathsf{form}(x) > \mathsf{form}(y) \,]$

Why are we unable to derive the other theorems? Let us examine in detail one of Hage's theorems that we cannot derive. The first theorem discussed in the text is *the higher the centralization, the higher the stratification* (C.19), supposedly derivable using A.1 and A.5 (p.299/300). In our formal set-up this theorem, T.19, would read:

$$\forall x, y \, [\, \mathsf{cent}(x) < \mathsf{cent}(y) \rightarrow \mathsf{stra}(x) < \mathsf{stra}(y) \,]$$

OTTER cannot derive this theorem (neither from F.1 and F.5, nor from the total set of axioms). Since we fail to prove this conjecture, we can attempt to disprove it.

We use the automated model generator to look for counterexamples, that is, models in which the axioms hold, but the conjecture is falsified. Using MACE we can construct counterexamples to T.19 using input-file (w.l.o.g., we use only F.1 and F.5):

[6]The converse of T.10, $\forall x, y \, [\, \mathsf{comp}(x) < \mathsf{comp}(y) \rightarrow \mathsf{effi}(x) > \mathsf{effi}(y) \,]$, is also derivable.

```
set(auto).
formula_list(usable).
% F.1
all x y (cent(x)<cent(y) -> prod(x)<prod(y)).
% F.5
all x y (stra(x)<stra(y) -> prod(x)<prod(y)).
% negation of T.19
-( all x y (cent(x)<cent(y) -> stra(x)<stra(y)) ).
end_of_list.
```

Note that both OTTER (when trying to prove a conjecture) and MACE (when trying to disprove it) use exactly the same input-file!

After running MACE using a domain size 2 (for example -n2 -p -m10) we find the four counterexamples of Table 5. In the counterexamples, F.1 is satisfied, F.5 holds vacuously, but the claimed theorem, T.19, is falsified. This proves that the claimed theorem is not derivable. Is the claimed theorem a false conjecture? Or has something gone wrong when we translated the natural language axioms into first-order logic?

Table 5: Counterexamples To **T.19**.

	cent	prod	stra
O_1	0	0	0
O_2	1	1	0
O_1	0	0	1
O_2	1	1	1
O_1	1	1	0
O_2	0	0	0
O_1	1	1	1
O_2	0	0	1

3.3 A SECOND FORMALIZATION

From a logical perspective, three options present themselves:

1. Discard the intended theorem as a false conjecture.
2. Rescue the intended theorem by weakening it sufficiently such that it becomes derivable.
3. Rescue the intended theorem by qualifying these models as an unintended one, and strengthening the premises such that these models are excluded.

Option 1 basically means that we stick to the formalization F.1 through F.7 and limit the explanatory power of the theory from 21 'theorems' to just the three theorems T.2, T.6, and T.7. This seems like an outcome we would like to avoid.

For option 2 we need to transform the counterexamples to T.19 into examples, i.e., we have to weaken the intended theorem such that it does hold for the counterexamples. Let us analyze the counterexamples more precisely. They have the following form (let O_a denote O_1 in the first two models and O_2 in the second two):

$$\begin{aligned} \mathsf{cent}(O_a) < \mathsf{cent}(O_b) \quad \wedge \\ \mathsf{prod}(O_a) < \mathsf{prod}(O_b) \quad \wedge \\ \mathsf{stra}(O_a) = \mathsf{stra}(O_b) \end{aligned}$$

An obvious way to implement option 2 is to formalize the intended theorem as the weaker *the higher the centralization, the higher* ***or equal*** *the stratification*:

$$\forall x, y\ [\ \mathsf{cent}(x) < \mathsf{cent}(y)\ \rightarrow\ \mathsf{stra}(x) \leq \mathsf{stra}(y)\ ]$$

This weaker version of the theorem holds in the models of Table 5, turning the former counterexamples into examples. Now, we make a second attempt at proving this (weaker version of the) theorem using OTTER. Note that, although we have dealt with the (type of) counterexamples in Table 5, this gives no guarantee that there are no other counterexamples. There turns out to be none, because OTTER can prove the weaker version of T.19.[7]

This same strategy also works for T.10 and T.13, but not for the remaining other 15 claimed theorems. Consider for example the first conjecture, T.1:

$$\forall x, y\ [\ \mathsf{form}(x) < \mathsf{form}(y)\ \rightarrow\ \mathsf{prod}(x) < \mathsf{prod}(y)\ ]$$

MACE generates 12 counterexamples (cardinality 2, w.l.o.g. using only F.1 and F.3), two of which are listed in Table 6.

Table 6: Counterexamples To **T.1**.

	cent	form	prod
O_1	0	0	1
O_2	0	1	0
O_1	0	0	0
O_2	0	1	0

These counterexamples have the following forms:

- $\mathsf{form}(O_1) < \mathsf{form}(O_2)\ \wedge\ \mathsf{prod}(O_1) > \mathsf{prod}(O_2)$
- $\mathsf{form}(O_1) < \mathsf{form}(O_2)\ \wedge\ \mathsf{prod}(O_1) = \mathsf{prod}(O_2)$

Moreover, there are also models (of F.1 and F.3) in which the theorem does hold (note that these are no counterexamples but examples). These have the form:

- $\mathsf{form}(O_1) < \mathsf{form}(O_2)\ \wedge\ \mathsf{prod}(O_1) < \mathsf{prod}(O_2)$

There is no relation between the variables of formalization and production: a weaker version of T.1 that holds in all these models, will be a tautology. Pursuing option 2 gives us three additional theorems, i.e., the weaker versions of T.10, T.13 and T.19. Although this doubles the explanatory power of the theory, it remains somewhat doubtful that Hage did overlook counterexamples to the remaining 15 of his 21 theorems.

It may be more reasonable to assume that these counterexamples were not among the models that Hage intended for his theory (option 3). Based on our analysis above, a way to exclude the models that are counterexamples to T.19 is to add the axiom that a higher *production* will imply a higher *stratification* (the converse of F.5):

$$\forall x, y\ [\ \mathsf{prod}(x) < \mathsf{prod}(y)\ \rightarrow\ \mathsf{stra}(x) < \mathsf{stra}(y)\ ]$$

After adding this axiom, the models of Table 5 are no longer models of the (modified) theory, making these counterexamples disappear. This is confirmed by OTTER which can now prove the theorem T.19.

One way to view this revision is as adding an axiom to the premise set, but there's another way to view it. We can combine both F.5 and its converse to form F.5′, a revised formalization of axiom A.5:

$$\forall x, y\ [\ \mathsf{stra}(x) < \mathsf{stra}(y)\ \leftrightarrow\ \mathsf{prod}(x) < \mathsf{prod}(y)\ ]$$

As a result, we have reformalized the natural language axiom *the higher the stratification, the higher the production* by interpreting it as a bi-implication. This interpretation can be justified considering the ambiguity of the natural language axioms.

This strategy works for all the theorems in (Hage 1965) that were not derivable from F.1 through F.7, causing similar revisions to the other axioms.[8] Table 7 contains the revised first-order formalization of the axioms. Now that we have interpreted all of Hage's axioms as bi-implications we can, using OTTER, derive the all the corollaries that are mentioned in Table 1. Not only can we derive the conditional version of the corollaries, but we can also derive the 21 corresponding bi-implications. For example, OTTER can derive

[7]OTTER requires an axiom expressing that the ordering is strict, for example $\forall x, y\ \neg[\ x < y\ \wedge\ y < x\ ]$. In MACE the order "<" is build-in.

[8]The converse of F.1 (because of T.1, 8-9, 13, 16), F.2 (T.10, 17, 19), F.3 (T.10, 17-18, 20-21), F.4 (T.3-4), F.5 (T.5, 11-12, 14-15, 17-21), and F.7 (T.10, 13) are necessary for the derivation of theorems. The converse of F.6 is not! We decide to include the converse of F.6 in order to give similar interpretations of all natural language axioms.

Table 7: The Revised Formal Assumptions.

F.1′	$\forall x, y$ [$\mathsf{cent}(x) < \mathsf{cent}(y) \leftrightarrow \mathsf{prod}(x) < \mathsf{prod}(y)$]
F.2′	$\forall x, y$ [$\mathsf{form}(x) < \mathsf{form}(y) \leftrightarrow \mathsf{effi}(x) < \mathsf{effi}(y)$]
F.3′	$\forall x, y$ [$\mathsf{cent}(x) < \mathsf{cent}(y) \leftrightarrow \mathsf{form}(x) < \mathsf{form}(y)$]
F.4′	$\forall x, y$ [$\mathsf{stra}(x) < \mathsf{stra}(y) \leftrightarrow \mathsf{jobs}(x) > \mathsf{jobs}(y)$]
F.5′	$\forall x, y$ [$\mathsf{stra}(x) < \mathsf{stra}(y) \leftrightarrow \mathsf{prod}(x) < \mathsf{prod}(y)$]
F.6′	$\forall x, y$ [$\mathsf{stra}(x) < \mathsf{stra}(y) \leftrightarrow \mathsf{adap}(x) > \mathsf{adap}(y)$]
F.7′	$\forall x, y$ [$\mathsf{comp}(x) < \mathsf{comp}(y) \leftrightarrow \mathsf{cent}(x) > \mathsf{cent}(y)$]

T.19′:

$$\forall x, y\ [\ \mathsf{cent}(x) < \mathsf{cent}(y) \leftrightarrow \mathsf{stra}(x) < \mathsf{stra}(y)\ ]$$

Also the theorems, formulated in the same way as the axioms, can be interpreted as bi-implications. This result provides some confidence for the interpretation in Table 7 (and for the choice to regard the counterexamples as non-intended models of the theory).[9]

Using F.1′ through F.5′, MACE generates 258 models on a domain of size 2 (one of which is shown in Table 8). Consequently, the revised formalization is still a consistent theory. Inspecting this model, we can confirm that the theorems are satisfiable; for example, it is a model of theorem T.19′. We can also check whether theorems are falsifiable by constructing a model in which the theorem does not hold (disregarding the premises). For example, the models in Table 5 still prove that theorem T.19′ is falsifiable. Note that these models are necessarily no longer models of the revised axioms, otherwise they would still be counterexamples to the theorem. Since T.19′ is both satisfiable and falsifiable, the theorem is contingent—it is neither a tautology nor a contradiction.

Table 8: A Model Of Assumptions **F.1′** Through **F.7′**.

	cent	prod	form	effi	stra	jobs	adap	comp
O_1	0	0	0	0	0	1	1	1
O_2	1	1	1	1	1	0	0	0

We can explore the theory's domain by examining its models. Interestingly, in 256 of the 258 models of F.1′ through F.5′ with cardinality 2, *all* axioms hold vacuously (because each function is equal for both elements of the domain). For each of the eight functions there are 2 options to be equal, both zero or both one, giving 256 ($= 2^8$) models. As soon as one of the functions is unequal, all functions are unequal. This results in a model as depicted in Table 8 (and the isomorphic copy with O_1 and O_2 interchanged being the remaining last model).

If one interprets the 0 as 'low' and the 1 as 'high' then this model represents to the 'two ideal types' of organization that are discussed on (Hage 1965, p.304-307). One 'ideal type' is an 'organic model' that has high complexity, low centralization, low formalization, low stratification, high adaptiveness, low production, low efficiency, and high job satisfaction (corresponding to O_1 in the model of Table 8). This organization type emphasizes adaptiveness. The other 'ideal type' is a 'mechanistic model'. This opposite of the 'organic model' has low complexity, high centralization, high formalization, high stratification, low adaptiveness, high production, high efficiency, and low job satisfaction (corresponding to O_2 in the model of Table 8). This organization type emphasizes production.

3.4 SUMMARIZING

In order to derive the claimed theorems, we had to interpret the natural language assumptions as logical bi-implications. On the one hand, this rescues the theory: the intended theorems are soundly derivable, the theorems are contingent (both satisfiable and falsifiable), and the theory is consistent. On the other hand, this trivializes the theory: now all eight functions become indistinguishable.[10] In short, one has to conclude that the axioms of (Hage 1965) are too strong. Nevertheless, the attempt that Hage undertook, i.e., to construct a general, axiomatic theory of organizations, remains an important enterprise. Any effort to formulate some of the basic axioms of organization theory, should take Hage's attempt into account.

A critical analysis of Hage's theory is unfair without noting the incomparability of his and our positions. Science has progressed significantly since the sixties: much more is known about formal logics and about their application; automated reasoning tools are operational and are valuable research assistants; and also the social sciences have advanced, e.g., empirical data have become available. This has led to renewed interest in building axiomatic, formal theories in the social sciences.

[9]Jerry Hage later confirmed that the axioms and theorems should indeed be interpreted as biconditionals, and mentioned that he explicitly included the words "and *vice versa*" in later references to the theory. Note that without adding the converse of F.6 the theorems T.4-5, 12, 15, 18, 21 can only be derived as conditionals.

[10]From a strictly formal point of view, one would need a trichotomy axiom for each of the eight functions to ensure that values for each function are comparable. For example, $\forall x, y$ [$\mathsf{cent}(x) < \mathsf{cent}(y) \lor \mathsf{cent}(x) = \mathsf{cent}(y) \lor \mathsf{cent}(x) > \mathsf{cent}(y)$].

Table 9: Theoretical Criteria And Automated Reasoning Tools.

CRITERION	THEOREM PROVER	MODEL GENERATOR
Consistency		×
Inconsistency	×	
Soundness	×	
Unsoundness		×
Falsifiability		×
Unfalsifiability	×	
Contingency		×
Noncontingency	×	
Domain		×

4 CONCLUSIONS AND DISCUSSION

In this paper we outlined a theory building methodology that is based on the use of standard first-order logic, and of existing automated reasoning tools. The logic provides us with a number of criteria that can be tested for using computational tools, such as consistency, soundness, falsifiability, and contingency (see Table 9). In principle, each criterion can be tested for by both theorem proving and model generation strategies, for example, a theorem is also sound if it holds in all models of the premise set, or a theory is consistent if the deductive closure of the premise set does not contain a contradiction. In practice, the use of automated theorem provers and model generators is complementary: generating all models or the complete deductive closure of a premise set is impossible. A theorem prover is suitable for proving the inconsistency of the theory, or the soundness of a derivation (requiring only a single proof), and a model generator can prove the consistency of the theory, or the unsoundness of a conjecture (requiring only a single model). In short, much is to be gained by using the right tool for the specific proof/disproof attempt at hand, and even more than just computational differences. Consider, for example, a situation in which the prover fails to prove a conjecture. Determining what caused this failure typically requires a thorough examination of the search-traces—an arduous, time-consuming activity. If the model generator can construct a counterexample to the conjecture, it will become apparent immediately why the proof attempt failed.

As always, there are principal and practical limitations to use of automated reasoning tools: first-order logic is not decidable (although it is semi-decidable: it may detect a consequent eventually); current automated model generators can only find finite models (even only very small ones, cardinalities beyond a dozen seem impractical); and the common practical limitations such as memory, CPU, time. However, none of the proofs and models searches for the case study in Section 3 requires more than five seconds. Admittedly, this case study concerns a relatively simple theory fragment. Larger theories have been formalized in some of the other case studies (for example (Péli and Masuch 1997) where proofs required up to 30 minutes). Current implementations of automated theorem provers, including OTTER, are very powerful. Automated model generators are of recent incarnation, and are yet far less sophisticated. MACE chokes on deeply nested terms or clauses with many literals (beyond 10 distinct terms). We might end up in a situation in which we cannot prove a conjecture, nor find small counterexamples to it (for example, when all counterexamples have infinite cardinality). Still, current automated model generators are powerful enough to have solved several open problems in (finite) mathematics (Slaney 1994).

We started this paper by referring to the positivist heritage that we share—the logical analysis for evaluating and justifying scientific theories. However, the use of the tools goes beyond a rigid, final justification of theories. We use them extensively during the process of formalization. As a result, we also enter the context of discovery: during the formalization process, we will repeatedly refine the (formal) theory. This makes our methodology more in line with more recent philosophy of science, in particular with (Lakatos 1976). Lakatos gives a logical analysis of the development of theories over time, with which the case study of section 3 shows remarkable resemblance (especially his classroom dialogue attempting to prove the *polyhedron* conjecture).

In our experience, the tools are especially useful during the process of formalizing a theory—intermediate versions of a theory under construction are more likely to contain logical flaws. When formalizing a larger theory, a short lapse of attention may result in an inconsistent theory. Proving consistency by generating a model can be a fast and easy safeguard against such an unfortunate event. Moreover, on many occasions (especially in the social sciences), theorems cannot be derived because some background knowledge is missing. The need for such an assumption can become clear immediately by examining the counterexamples. It is often difficult to find such non-intended models by hand, because they conflict with our common sense.

Acknowledgements

Thank you to Jeroen Bruggeman, Jerry Hage, Michael Masuch, Breanndán Ó Nualláin, and László Pólos. This research was in part supported by the Netherlands Organization for Scientific Research (NWO grant # PGS 50-334, # 030-50-334, and # SIR 12-2918). Part of the material in section 3 was presented at the IJCAI'97 Workshop on Model-Based Automated Reasoning. Address correspondence to: Jaap Kamps, Applied Logic Laboratory, Universiteit van Amsterdam, Sarphatistraat 143, 1018 GD Amsterdam, The Netherlands, Tel. +31-20 5252538, Fax. +31-20 5252800, Email: ⟨kamps@ccsom.uva.nl⟩.

References

Adorno, T. W., R. Dahrendorf, H. Pilot, H. Albert, J. Habermas, and K. R. Popper (1969). *Der Positivismusstreit in der deutchen Soziologie.* Hermann Luchterhand Verlag, Darmstadt und Neuwied.

Ayer, A. J. (Ed.) (1959). *Logical Positivism*, The library of philosophical movements. The Free Press, New York.

Bruggeman, J. (1997). Niche width theory reappraised. *Journal of Mathematical Sociology 22*(2), 201–220.

Bundy, A. (Ed.) (1994). *Automated Deduction — CADE-12*, Volume 814 of *Lecture Notes in Artificial Intelligence.* Springer-Verlag, Berlin.

Glorie, J. C., M. Masuch, and M. Marx (1990). Formalizing organizational theory: A knowledge-based approach. In M. Masuch (Ed.), *Organization, Management, and Expert Systems: models of automated reasoning*, Chapter 4, pp. 79–81. De Gruyter, Berlin, New York.

Hage, J. (1965). An axiomatic theory of organizations. *Administrative Science Quarterly 10*(3), 289–320.

Hannan, M. T. (1997). Rethinking age dependence in organizational mortality: Logical formalizations. Technical report, Stanford University.

Kamps, J. and L. Pólos (1998). Reducing uncertainty: A formal theory of *organizations in action.* CCSOM Working Paper 98-167.

Lakatos, I. (1976). *Proofs and Refutations. The logic of mathematical discovery.* Cambridge University Press, Cambridge, England.

McCune, W. (1994a). A Davis-Putnam program and its application to finite first-order model search: Quasigroup existence problems. Technical report, Argonne National Laboratory, Argonne IL. DRAFT.

McCune, W. (1994b). OTTER: Reference manual and guide. Technical Report ANL-94/6, Argonne National Laboratory, Argonne IL.

Péli, G. (1997). The niche hiker's guide to population ecology: A logical reconstruction of organizational ecology's niche theory. In A. E. Raftery (Ed.), *Sociological Methodology 1997*, pp. 1–46. Blackwell, Oxford UK.

Péli, G., J. Bruggeman, M. Masuch, and B. Ó Nualláin (1994). A logical approach to formalizing organizational ecology. *American Sociological Review 59*(4), 571–593.

Péli, G. and M. Masuch (1997). The logic of propagation strategies: Axiomatizing a fragment of organizational ecology in first-order logic. *Organization Science 8*(3), 310–331.

Popper, K. R. (1959). *The Logic of Scientific Discovery.* Hutchinson, London.

Popper, K. R. (1969). Die Logik der Sozialwissenschaften. See Adorno, Dahrendorf, Pilot, Albert, Habermas, and Popper (1969), pp. 103–123.

Rudner, R. S. (1966). *Philosophy of Social Science.* Foundations of Philosophy Series. Prentice-Hall, Englewood Cliffs NJ.

Slaney, J. (1994). The crisis in finite mathematics: Automated reasoning as cause and cure. See Bundy (1994), pp. 1–13.

Suppes, P. (1968). The desirability of formalization in science. *Journal of Philosophy LXV*(20), 651–664.

Sutcliffe, G., C. Suttner, and T. Yemenis (1994). The TPTP problem library. See Bundy (1994), pp. 252–266.

Tarski, A. (1956). *Logic, Semantics, Metamathematics.* Oxford University Press, New York.

On the logic of merging

Sébastien Konieczny **Ramón Pino-Pérez**
Laboratoire d'Informatique Fondamentale de Lille URA 369 CNRS
Université de Lille I - 59655 Villeneuve d'Ascq - FRANCE
{konieczn,pino}@lifl.fr

Abstract

This work proposes an axiomatic characterization of merging operators. It underlines the differences between arbitration operators and majority operators. A representation theorem is stated showing that each merging operator corresponds to a family of partial pre-orders on interpretations. Examples of operators are given. They show the consistency of the axiomatic characterization. A new merging operator $\triangle_{GMax}$ is provided. It is proved that it is actually an arbitration operator.

1 Introduction

In a growing number of applications, we face conflicting information coming from several sources. The problem is to reach a coherent piece of information from these contradicting ones. A lot of different merging methods have already been given [BI84, LMa, BKM91, BKMS92, Sub94]. Instead of giving one particular merging method we propose, in this paper, a characterization of such methods following the rationality of the postulates they satisfy. We shall call merging operators those methods that obey a minimal set of rational merging postulates. Then we shall investigate two subclasses of merging operators: arbitration operators and majority operators.

Merging operators are useful in a lot of applications: to find a coherent information in a distributed database system, to solve a conflict between several people or several agents, to find an answer in a decision-making committee, to take a decision when information given by some captors is contradictory, etc.

This work is related to the AGM (Alchourrón, Gärdenfors, Makinson) framework of revision theory [AGM85, Gär88, KM91]. Revision is the process of according a knowledge base in the view of a new evidence. One basic assumption of revision is that the new information is more reliable that the knowledge base, but it is not always the case. We can distinguish 3 cases:

- *The new piece of information is more reliable than the knowledge base:* it is the assumption made in the revision theory so we can revise our knowledge base by the new piece of information.
- *The new piece of information is less reliable than the knowledge base:* a drastic point of view could be to ignore this unreliable piece of information but if we want to be more constructive we can take this piece of information into account if it is consistent with the knowledge base and ignore it only if it is inconsistent with our belief. Another interesting way would be to reverse the revision, i.e. to revise the new piece of information by the knowledge base.
- *The new piece of information is as reliable as the knowledge base:* here we can't give the preference to one of the two items of knowledge, so we have to find something else. This is the aim of merging operators.

The intuitive difference between arbitration and majority operators is that arbitration operators reach a consensus between the protagonists' views by trying to satisfy as much as possible all the protagonists, whereas majority operators elect, in a sense, the result of the merging by taking the majority into account. In other words arbitration operators try to minimize individual dissatisfaction, whereas majority operators try to minimize global dissatisfaction. One of our main concerns in this work is to state these intuitions in a formal way.

Some operators quite close to merging operators have already been formally studied. Revesz defined in [Rev93, Rev97] model-fitting operators which can be considered as a generalization of revision for multiple knowledge bases. Revesz also defined arbitration operators from model-fitting operators. We make a criticism about Revesz's postulates: they do not distinguish between majority and arbitration.

Liberatore and Schaerf have proposed postulates to characterize arbitration [LS95, LS98]. Their definition has a strong connection with revision operators, but the major drawback, in our opinion, is that those operators arbitrate only two knowledge bases. Furthermore they select some interpretations in the two knowledge bases as the result of the arbitration. We consider that we can't ignore interpretations which do not belong to these knowledge bases, consider the following example:

Example 1 Suppose that we want to speculate on the stock exchange. We ask two financial experts about four shares A,B,C,D. We denote 1 if the share rises and 0 if it falls (we suppose that its value can't be stable). These agents have the same expert level and so they are both equally reliable. The first one says that all the shares will rise: $\varphi_1 = \{(1,1,1,1)\}$, the second one thinks that all the shares will fall: $\varphi_2 = \{(0,0,0,0)\}$. The Liberatore and Schaerf operators will arbitrate these opinions and give the following result: $R = \{(0,0,0,0),(1,1,1,1)\}$. So it means that either φ_1 is totally wrong or it's φ_2 who is completely mistaken. But intuitively, if the two experts are equally reliable, there is no reason to think that one of them has failed more than the other: they both have to be at the same "distance" of the truth. So they are certainly both wrong on two shares and the result has to be: $R' = \{(0,0,1,1),(0,1,0,1),(0,1,1,0),(1,0,0,1),(1,0,1,0),(1,1,0,0)\}$. So two of the shares will rise and two will fall but we don't know which ones.

In our opinion Liberatore and Schaerf's operators have to be seen as selection operators and have to be used in applications which require the result be one of the possibilities given by the protagonists. For example, if the result of the arbitration is a medical treatment, we can't "merge" several therapies and so we have to use Liberatore and Schaerf operators. Liberatore and Schaerf's operators take, in a sense, the interpretation as unit of change, we propose to take the propositional variable as such a unit, as Dalal says in [Dal88]: *"Change in truth value of a single symbol can be considered as the smallest unit of change"*, we want to apply this to arbitration.

Lin and Mendelzon proposed a *theory merging by majority* operator [LMa, Lin96] which solves conflicts between knowledge bases by taking the majority into account. Their *theory merging operators* are what we call majority operators.

The paper is organized as follows: in section 2 we give some definitions and state some notations. In section 3 we propose postulates for merging operators, majority operators and arbitration operators and we study the relationships between some of the postulates. In section 4 we give a model-theoretic characterization of those operators. In section 5 we give some examples of merging operators, especially we show that an operator, called $\triangle_{GMax}$, is an arbitration operator. Finally, in section 6 we give some conclusions and discuss open problems.

2 Preliminaries

We consider a propositional language $\mathcal{L}$ over a finite alphabet $\mathcal{P}$ of propositional letters. 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, $Mod(\varphi)$ denote the set of models of φ. And let M be a set of interpretations, $form(M)$ denote a formula which set of models is M. When $M = \{I\}$ we will use the notation $form(I)$ for reading convenience.

A *knowledge base* K is a finite set of propositional formulae which can be seen as the formula φ which is the conjunction of the formulae of K. By abuse, we will use K to denote the formula φ. We will note K_I a knowledge base the sole model is I.

Let $K_1, \dots, K_n$ be n knowledge bases (not necessarily different). We call *knowledge set* the multi-set E consisting of those n knowledge bases: $E = \{K_1, \dots, K_n\}$. We note $\bigwedge E$ the conjunction of the knowledge bases of E, i.e. $\bigwedge E = K_1 \wedge \dots \wedge K_n$. The union of multi-sets will be noted $\sqcup$.

Remark 2 *Since an inconsistent knowledge base gives no information for the merging process, we'll suppose in the rest of the paper that the knowledge bases are consistent.*

$\mathcal{K}$ will denote the set of consistent knowledge bases and $\mathcal{E}$ will denote the set of non empty finite multi-sets with elements in $\mathcal{K}$.

Let's denote $\mathcal{S}$ the set of sets of interpretations without the empty set, *i.e.* $\mathcal{S} = \mathcal{P}(\mathcal{W}) \setminus \{\emptyset\}$; and let's denote $\mathcal{M}$ the set of finite non empty multi-sets with elements in $\mathcal{S}$. Elements of $\mathcal{S}$ and $\mathcal{M}$ will be denoted by the letters S and M respectively with possibly subscripts.

So a typical element $M \in \mathcal{M}$ will be of the shape $\{S_1, \ldots, S_n\}$. Let $M = \{S_1, \ldots, S_n\}$, we define $\bigcap M$ in the usual way: $I \in \bigcap M$ iff $\forall S_i \in M \quad I \in S_i$.

Definition 3 *A knowledge set E is consistent if and only if $\bigwedge E$ is consistent. We will use $Mod(E)$ to denote $Mod(\bigwedge E)$.*

Definition 4 *Let E_1, E_2 be two knowledge sets. E_1 and E_2 are equivalent, noted $E_1 \leftrightarrow E_2$, iff there exists a bijection f from $E_1 = \{K_1^1, \ldots, K_n^1\}$ to $E_2 = \{K_1^2, \ldots, K_n^2\}$ such that $\vdash f(K) \leftrightarrow K$.*

Note that the relation $\leftrightarrow$ is an equivalence relation on knowledge sets. As usual, we denote by $\mathcal{E}/\leftrightarrow$ the quotient of $\mathcal{E}$ by the relation $\leftrightarrow$. Thus the function $\imath : \mathcal{E}/\leftrightarrow \longrightarrow \mathcal{M}$, defined by $\imath([\{K_1, \ldots, K_n\}]_\leftrightarrow) = \{Mod(K_1), \ldots, Mod(K_n)\}$ is a bijection. By abuse we will write $\imath(E)$ instead of $\imath([E]_\leftrightarrow)$.

A pre-order over $\mathcal{W}$ is a reflexive and transitive relation on $\mathcal{W}$. Let $\leq$ be a pre-order over $\mathcal{W}$, 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 $J \leq I$. Let I be an interpretation, we wrote $I \in min(\leq)$ iff $\nexists J \in \mathcal{W}$ s.t. $J < I$.

By abuse if R is in $\mathcal{K}$ (respectively in $\mathcal{S}$) then R will denote also the multi-set $\{R\}$ which is in $\mathcal{E}$ (resp. in $\mathcal{M}$). For a positive integer n we will denote R^n the multi-set $\{\underbrace{R, \ldots, R}_{n}\}$. Thus $R^n = \underbrace{R \sqcup \ldots \sqcup R}_{n}$.

An operator $\triangle$ will be a function mapping knowledge sets into knowledge bases. In the rest of the paper we will distinguish between *operator* and *merging operator*: the former when no special properties are satisfied the later to indicate that the operator satisfies the postulates of definition 5. Let K, E and $\triangle$ be a knowledge base, a knowledge set and an operator respectively. We define the sequence $\langle \triangle^n(E, K)\rangle_{n \geq 1}$ by the following:

$$\triangle^1(E, K) = \triangle(E \sqcup K)$$
$$\text{and} \quad \triangle^{n+1} = \triangle(\triangle^n(E, K) \sqcup K)$$

3 Postulates

In this section, we are going to propose a characterization of merging operators, i.e. we give a minimal set of properties an operator has to satisfy in order to have a rational behaviour concerning the merging. Let E be a knowledge set, and let $\triangle$ be an operator which assigns to each knowledge set E a knowledge base $\triangle(E)$.

Definition 5 *$\triangle$ is a merging operator if and only if it satisfies the following postulates:*

(A1) *$\triangle(E)$ is consistent*

(A2) *If E is consistent, then $\triangle(E) = \bigwedge E$*

(A3) *If $E_1 \leftrightarrow E_2$, then $\vdash \triangle(E_1) \leftrightarrow \triangle(E_2)$*

(A4) *If $K \wedge K'$ is not consistent, then $\triangle(K \sqcup K') \nvdash K$*

(A5) *$\triangle(E_1) \wedge \triangle(E_2) \vdash \triangle(E_1 \sqcup E_2)$*

(A6) *If $\triangle(E_1) \wedge \triangle(E_2)$ is consistent, then $\triangle(E_1 \sqcup E_2) \vdash \triangle(E_1) \wedge \triangle(E_2)$*

These six postulates are the basic properties a merging operator has to satisfy, the intuitive meaning of the postulates is easy to understand: we always want to extract a piece of information from the knowledge set, what is forced by $(A1)$ (Notice that, as assumed in remark 2, all the knowledge bases of the knowledge set are consistent). If all the knowledge bases agree on some alternatives, $(A2)$ assures that the result of the merging will be the conjunction of the knowledge bases. $(A3)$ states that the operator $\triangle$ obeys a principle of irrelevance of syntax, i.e. if two knowledge sets are equivalent in the sense of definition 4, then the two knowledge bases resulting from the merging will be logically equivalent. $(A4)$ is the fairness postulates, the point is that when we merge two knowledge bases, merging operators must not give preference to one of them. We will see (theorem 11) that (A4) is the clue for distinguishing arbitration operators from majority operators. $(A5)$ expresses the following idea: if a group E_1 compromises on a set of alternatives which I belongs to, and another group E_2 compromises on an another set of alternatives which contains I, so I has to be in the chosen alternatives if we join the two groups. $(A5)$ and $(A6)$ together state that if you could find two subgroups which agree on at least one alternative, then the result of the global arbitration will be exactly those alternatives the two groups agree on. The postulates $(A5)$ and $(A6)$ have been given in [Rev97] by Revesz for weighted model fitting operators.

Observation 6 *By definition, merging operators are commutative, i.e. the result of a merging does not depend on any order of elements of the knowledge set.*

Let's now turn our attention to the difference between majority and arbitration operators. We give here a postulate that renders the behaviour of majority operators, that is to say that if an opinion has a large audience, then it will be the opinion of the group:

$(M7) \quad \forall K\ \exists n\ \ \triangle(E \sqcup K^n) \vdash K$

Thus we define majority operators by the following:

Definition 7 *A merging operator is a majority operator if it satisfies* $(M7)$.

Besides, arbitration operators are those operators which are, in a large extent, majority insensitive. We first give a postulate which seems to be a good characterization of arbitration operator:

$$(A7') \quad \forall K\, \forall n \;\; \Delta(E \sqcup K^n) = \Delta(E \sqcup K)$$

This postulate states that the result of an arbitration is fully independent from the frequency of different views. Unfortunately the set of postulates $\{A1, \ldots, A6, A7'\}$ is not consistent. The proof of this result has been pointed out by P. Liberatore (personal communication):

Theorem 8 *There is no merging operator satisfying* $(A7')$.

Proof: Let $E_1 = \{K, \neg K\}$ and $E_2 = \{K\}$ be two knowledge sets. By $(A7')$ we have that $\Delta(E_1 \sqcup E_2) = \Delta(E_1)$. By $(A4)$ we have also that $\Delta(E_1) \nvdash K$ and $\Delta(E_1) \nvdash \neg K$. Furthermore by $(A2)$ we deduce $\Delta(E_2) = K$. So $\Delta(E_1) \wedge \Delta(E_2)$ is consistent and by $(A6)$ we have $\Delta(E_1 \sqcup E_2) \vdash \Delta(E_1) \wedge \Delta(E_2)$, it can be rewritten as $\Delta(E_1) \vdash \Delta(E_1) \wedge K$. Then $\Delta(E_1) \vdash K$, which contradicts $(A4)$. ∎

Thus if we want to have a postulate expressing majority insensitivity while being consistent with $(A1 - A6)$ we must weaken $(A7')$. We propose the following alternative:

$$(A7) \quad \forall K'\, \exists K\; K' \nvdash K\; \forall n\; \Delta(K' \sqcup K^n) = \Delta(K' \sqcup K)$$

$(A7)$ states that, to a large extent, the result of the arbitration is independent from the frequency of the different views.

And we define arbitration operator in the following way:

Definition 9 *A merging operator is an arbitration operator if it satisfies* $(A7)$.

Now we investigate some relations between the postulates.

Theorem 10 *If an operator satisfies* $(A1)$, *then it can't satisfy both* $(A7')$ *and* $(M7)$.

Proof: From $(A7')$ and $(M7)$ we deduce that for any arbitrary E

$$\forall\, K \;\; \Delta(E \sqcup K) \vdash K \quad (*)$$

Take K' such that $K \wedge K' \vdash \bot$. Now putting $E = K'$, by $(*)$, we have $\Delta(K' \sqcup K) \vdash K$. In a symmetrical way we have $\Delta(K \sqcup K') \vdash K'$ so $\Delta(K \sqcup K') \vdash K \wedge K'$ and then $\Delta(K \sqcup K') \vdash \bot$ which contradicts $(A1)$. ∎

A merging operator can't be an arbitration operator and a majority operator, more precisely we have the following:

Theorem 11 *If an operator satisfies* $(A4)$, *then it can't satisfy both* $(A7)$ *and* $(M7)$.

Proof: From $(A7)$ and $(M7)$ we deduce easily $\forall K'\; \exists K\; K' \nvdash K \;\; \Delta(K' \sqcup K) \vdash K$. Let's choose $K' = K_I = form(I)$, then $\exists K\; K_I \nvdash K \;\; \Delta(K_I \sqcup K) \vdash K$. But $K_I \nvdash K$ is equivalent to $K_I \wedge K \vdash \bot$ and so by $(A4)$ we have that $\Delta(K_I \sqcup K) \nvdash K$. Contradiction. ∎

So, although it seems very weak, the fairness postulate $(A4)$ play a very important role, since it allows us to differentiate arbitration operators and majority operators.

In addition to these basic postulates we can find various other properties, we investigate some of them below.

An interesting property for a merging operator is the following which we call the *iteration* property:

$$(A_{it}) \quad \exists n\; \Delta^n(E, K) \vdash K$$

The intuitive idea is that, since the merging operators give, in a sense, the average knowledge of a knowledge set, if we always take the result of a merging and iterate with the same knowledge base, we have to reach this knowledge base after enough iterations. But, even if it seems to be a reasonable requirement, we don't know if all merging operators obey (A_{it}), more exactly we suspect that those operators satisfying (A_{it}) are topological operators, *i.e.* operators defined from a distance.

Now let's turn our attention to the two properties of associativity and monotony. We claim that they are not desirable for merging operators and we show that merging operators do not satisfy any of them. First let's give a formal definition of associativity and monotony:

$$(Ass) \quad \Delta(E_1 \sqcup \Delta(E_2)) = \Delta(E_1 \sqcup E_2)$$

Associativity seems to be an interesting property since it would allow sub-merging within the knowledge set. So merging could be implemented more easily and more efficiently.

(Mon) If $K_1 \vdash K'_1, \ldots, K_n \vdash K'_n$ then $\Delta(K_1 \sqcup \ldots \sqcup K_n) \vdash \Delta(K'_1 \sqcup \ldots \sqcup K'_n)$

The monotony property expresses that if a knowledge set E_1 is "stronger" than a knowledge set E_2, then the merging of E_1 has to be logically stronger than the merging of E_2.

Theorem 12 *If an operator satisfies* $(A2)$ *and* $(A4)$*, then it doesn't satisfy* (Mon)*.*

Proof: Let I, J be two different interpretations. Let $K_1 = K'_1 = form(I)$, $K_2 = form(J)$, and $K'_2 = form(I, J)$, so we have $K_1 \vdash K'_1$ and $K_2 \vdash K'_2$. From $(A2)$ $\Delta(K'_1 \sqcup K'_2) = form(I)$ and from $(A4)$ $\Delta(K_1 \sqcup K_2) \nvdash form(I)$. So we have $\Delta(K_1 \sqcup K_2) \nvdash \Delta(K'_1 \sqcup K'_2)$. ∎

So it is clear that monotony is not satisfied by merging operators, it is not exactly the same with associativity, we show that it is not satisfied by majority operators and that it is not compatible with the iteration property:

Theorem 13 *If an operator satisfies* $(A2)$ $(A4)$ *and* $(M7)$*, then it can't satisfy* (Ass)*.*

Proof: Let's take K_I and K_J two different complete formulae, by $(M7)$ we have that $\exists n\, \Delta(K_I \sqcup K_J^n) \vdash K_J$. By (Ass) we have that $\Delta(K_I \sqcup K_J^n) = \Delta(K_I \sqcup \Delta(K_J^n))$. But by $(A2)$ we have $\Delta(K_J^n) = K_J$. So we obtain that $\Delta(K_I \sqcup K_J) \vdash K_J$. What contradicts $(A4)$. ∎

Theorem 14 *If an operator satisfies* $(A2)$ *and* $(A4)$*, then it can't satisfy both* (A_{it}) *and* (Ass)*.*

Proof: (A_{it}) $\exists n$ $\Delta^n(E, K) \vdash K$, but by (Ass) we find that $\Delta^n(E, K) = \Delta(E \sqcup K^n) = \Delta(E \sqcup \Delta(K^n))$ and by (A2) we have that $\Delta(E \sqcup \Delta(K^n)) = \Delta(E \sqcup K)$. So we have that $\Delta(E \sqcup K) \vdash K$, what, taking $E = K'$ with $K' \wedge K \vdash \perp$, contradicts (A4). ∎

So, if we want some additional property for a merging operator, we have to choose between iteration and associativity. We claim that iteration is a desirable property for merging operators, so associativity is not.

4 Semantical characterizations

In this section we give a model-theoretic characterization of merging operators first in terms of functions on sets of interpretations and then in terms of family of orders. More exactly we show that each merging operator corresponds to a function from multi-sets of sets of interpretations to sets of interpretations and then we show that each merging operator corresponds to a family of partial pre-orders on interpretations. The semantical characterization of the merging operators in terms of pre-orders is very close to the axiomatic characterization. This is due to the fact that we can't have a definition of the pre-order as subtle as in the case of belief revision. But this semantical characterization is very useful in the proofs and is a starting point for generalizing merging operators (*e.g.* when one considers the set of alternatives as a parameter).

First we define what is a merging function:

Definition 15 *A function* $\delta : \mathcal{M} \longrightarrow \mathcal{S}$ *is said to be a merging function if the following properties hold for any* $M, M_1, M_2 \in \mathcal{M}$ *and* $S, S' \in \mathcal{S}$*:*

1. *If* $I \in \bigcap M$*, then* $I \in \delta(M)$
2. *If* $\bigcap M \neq \emptyset$ *and* $I \notin \bigcap M$*, then* $I \notin \delta(M)$
3. *If* $S \cap S' = \emptyset$*, then* $\delta(S \sqcup S') \nsubseteq S$
4. *If* $I \in \delta(M_1)$ *and* $I \in \delta(M_2)$*, then* $I \in \delta(M_1 \sqcup M_2)$
5. *If* $\delta(M_1) \cap \delta(M_2) \neq \emptyset$ *and* $I \notin \delta(M_1)$*, then* $I \notin \delta(M_1 \sqcup M_2)$

A majority merging function is a merging function that satisfies the following:

6. $\forall M \in \mathcal{M}\ \forall S \in \mathcal{S}\ \exists n\ \delta(M \sqcup S^n) \subseteq S$

A fair merging function is a merging function that satisfies the following:

7. $\forall S' \in \mathcal{S}\ \exists S \in \mathcal{S}\ S' \nsubseteq S\ \forall n\ \delta(S' \sqcup S^n) = \delta(S' \sqcup S)$

It is easy to see, via the bijection $\imath$ of section 2 that the properties $1-5$ are the semantical counterparts of postulates $(A1-A6)$ (notice that postulate (A_1) corresponds to the fact $\emptyset \notin \mathcal{S}$), property 6 corresponds to postulate $(M7)$ and property 7 corresponds to postulate $(A7)$. More precisely we have the following representation theorem which proof is straightforward:

Theorem 16 *An operator* Δ *is a merging operator (it satisfies* $(A1-A6)$*) if and only if there exists a merging function* $\delta : \mathcal{M} \longrightarrow \mathcal{S}$ *such that*

$$Mod(\Delta(E)) = \delta(\imath(E)).$$

Furthermore Δ *is a majority merging operator iff* δ *is a majority merging function; and* Δ *is an arbitration operator iff* δ *is a fair merging function.*

As in the AGM framework for revision, we can suppose the existence of some relation which intuitively represents how credible each interpretation is for some given knowledge set. We will see that there is a close relationship between merging function and these relations on knowledge sets. First we define what a syncretic assignment is:

Definition 17 *A syncretic assignment is an assignment which maps each knowledge set E to a pre-order $\leq_E$ over interpretations such that for any $E, E_1, E_2 \in \mathcal{E}$ and for any $K, K' \in \mathcal{K}$:*

1. *If $I \in Mod(E)$ and $J \in Mod(E)$, then $I \simeq_E J$*
2. *If $I \in Mod(E)$ and $J \notin Mod(E)$, then $I <_E J$*
3. *If $E_1 \leftrightarrow E_2$, then $\leq_{E_1} = \leq_{E_2}$*
4. *If $Mod(K) \cap Mod(K') = \emptyset$, then $\min(\leq_{K \sqcup K'}) \not\subseteq Mod(K)$*
5. *If $I \in \min(\leq_{E_1})$ and $I \in \min(\leq_{E_2})$, then $I \in \min(\leq_{E_1 \sqcup E_2})$*
6. *If $\min(\leq_{E_1}) \cap \min(\leq_{E_2}) \neq \emptyset$ and $I \notin \min(\leq_{E_1})$, then $I \notin \min(\leq_{E_1 \sqcup E_2})$*

A majority syncretic assignment is a syncretic assignment which satisfies the following:

7. *$\forall E \in \mathcal{E}\ \forall K \in \mathcal{K}\ \exists n\ \min(\leq_{E \sqcup K^n}) \subseteq Mod(K)$*

A fair syncretic assignment is a syncretic assignment which satisfies the following:

8. *$\forall K'\ \ \exists K$ if $Mod(K') \not\subseteq Mod(K)$, then $\forall n\ \min(\leq_{K' \sqcup K^n}) = \min(\leq_{K' \sqcup K})$*

If we have an assignment that maps each knowledge set E to a pre-order $\leq_E$ on $\mathcal{W}$, then we can define a function $\delta : \mathcal{M} \longrightarrow \mathcal{S}$ by the following: let $M \in \mathcal{M}$ and let $E \in \mathcal{E}$ be such that $\imath(E) = M$, put

$$\delta(M) \quad = \quad \min(\leq_E) \qquad (1)$$

If the assignment satisfies property 3 above then δ is well defined.

Conversely, if we have a function $\delta : \mathcal{M} \longrightarrow \mathcal{S}$ we can define a corresponding family of relations on interpretations as $\forall E \in \mathcal{E}$:

$$\leq_E = [\delta(\imath(E)) \times (\mathcal{W} \setminus \delta(\imath(E)))] \cup \{\{I, I\} \setminus I \in \mathcal{W}\} \quad (2)$$

It is easy to show that if we have a (majority, fair) syncretic assignment, then the merging function obtained by equation 1 is a (majority, fair) merging function. Conversely, if we have a (majority, fair) merging function, then the family of relations obtained by equation 2 is a (majority, fair) syncretic assignment. This observation together with theorem 16 gives us straightforwardly the following:

Theorem 18 *An operator is a merging operator (respectively majority merging operator or arbitration operator) if and only if there exists a syncretic assignment (respectively majority syncretic assignment or fair syncretic assignment) that maps each knowledge set E to a pre-order $\leq_E$ such that*

$$Mod(\Delta(E)) = \min(\leq_E).$$

As pointed out by D. Makinson (personal communication), this definition of merging operators from such assignments can be compared to the framework of social choice theory [Kel78, Arr63]. The aim of social choice theory is to aggregate individual choices into a social choice, *i.e.* to find, for a given set of agents (corresponding to our knowledge sets) with individual preference relations, a social preference relation which reflects the preferences of the set of agents. This allows the definition of a welfare function selecting from a set of alternatives those that best fit the social preference relation.

5 Some merging operators

In this section we show the consistency of our merging postulates by giving three examples of operators. The first one is not a merging operator but it illustrates an approach to arbitration operators. The second one is a majority merging operator and the last one is a true arbitration operator.

For the following operators we will use the Dalal's distance [Dal88] to calculate the distance between two interpretations: let I, J be interpretations, $dist(I, J)$ is the number of propositional letters the two interpretations differ.

We also define the distance between an interpretation and a knowledge base as the minimum distance between this interpretation and the models of the knowledge base, that is:

$$dist(I, \varphi) = \min_{J \in Mod(\varphi)} dist(I, J)$$

Finally we define the distance between two knowledge bases by the following:

$$dist(\varphi, \varphi') = \min_{I \in Mod(\varphi)\ J \in Mod(\varphi')} dist(I, J)$$

Table 1: Distances

	φ_1	φ_2	φ_3	$\mathbf{dist_{Max}}$	$\mathbf{dist_{\Sigma}}$	$\mathbf{dist_{GMax}}$
(0,0,0)	1	1	3	3	5	(3,1,1)
(0,0,1)	0	0	2	2	2	(2,0,0)
(0,1,0)	2	0	2	2	4	(2,2,2)
(0,1,1)	1	1	1	1	3	(1,1,1)
(1,0,0)	0	2	2	2	4	(2,2,0)
(1,0,1)	0	1	1	1	2	(1,1,0)
(1,1,0)	1	1	1	1	3	(1,1,1)
(1,1,1)	1	2	0	2	3	(2,1,0)

The first operator we consider is the Δ_{Max} operator. It comes from an example of model fitting operator given by Revesz in [Rev97]. It is close to the minimax rule used in decision theory [Sav71]. The idea is to find the closest information to the overall knowledge set. Therefore it seems to be a good arbitration operator. But, as we will see, it doesn't satisfy all the postulates.

Definition 19 *Let φ be a knowledge base and E be a knowledge set:*

$$dist_{Max}(I,E) = \max_{\varphi \in E} dist(I,\varphi)$$

So, we define the following order:

$$I \leq_E^{Max} J \text{ iff } dist_{Max}(I,E) \leq dist_{Max}(J,E)$$

$$\text{and} \qquad Mod(\Delta_{Max}(E)) = \min(\leq_E^{Max})$$

The second operator we consider is the Δ_Σ operator. This is a majority merging operator as we will see below. Lin and Mendelzon give it as an example of what they called operators of *theory merging by majority* in [LMa]. Independently Revesz gives it as an example of weighted model fitting in [Rev93]. The Σ operator comes from a natural idea: the distance between an interpretation and a knowledge set is the sum of the distances between this interpretation and the knowledge bases of the knowledge set.

Definition 20 *Let E be a knowledge set and let I be an interpretation we put:*

$$dist_\Sigma(I,E) = \sum_{\varphi \in E} dist(I,\varphi)$$

$$I \leq_E^\Sigma J \text{ iff } dist_\Sigma(I,E) \leq dist_\Sigma(J,E)$$

$$\text{and} \qquad Mod(\Delta_\Sigma(E)) = min(\leq_E^\Sigma)$$

Next we present a new merging operator: Δ_{GMax} (stands for *Generalized* Max). The operator Δ_{GMax} is an arbitration operator and is a refinement of the Δ_{Max} operator.

Definition 21 *Let E be a knowledge set. Suppose $E = \{\varphi_1, \ldots, \varphi_n\}$. For each interpretation I we build the list $(d_1^I \ldots d_n^I)$ of distances between this interpretation and the n knowledge bases in E,* i.e. *$d_j^I = dist(I,\varphi_j)$. Let L_I be the list obtained from $(d_1^I \ldots d_n^I)$ by sorting it in descending order. Define $dist_{GMax}(I,E) = L_I$. Let $\leq_{lex}$ be the lexicographical order between sequences of integers. Now we put:*

$$I \leq_E^{GMax} J \text{ iff } dist_{GMax}(I,E) \leq_{lex} dist_{GMax}(J,E)$$

$$\text{and} \qquad Mod(\Delta_{GMax}(E)) = \min(\leq_E^{GMax})$$

We will illustrate the behaviour of these three operators on the database class example given by Revesz in [Rev93]:

Example 22 Consider a database class with three students: $E = \{\varphi_1, \varphi_2, \varphi_3\}$. The teacher can teach SQL, Datalog and O_2. He asks his students in turn to choose what to teach to satisfy the class best. The first student wants to learn SQL or O_2: $\varphi_1 = (S \vee O) \wedge \neg D$. The second wants to learn Datalog or O_2 but not both: $\varphi_2 = (\neg S \wedge D \wedge \neg O) \vee (\neg S \wedge \neg D \wedge O)$. The third wants to learn the three languages: $\varphi_3 = (S \wedge D \wedge O)$. Considering the propositional letters S, D and O in that order we have: $Mod(\varphi_1) = \{(1,0,0),(0,0,1),(1,0,1)\}$, $Mod(\varphi_2) = \{(0,1,0),(0,0,1)\}$, $Mod(\varphi_3) = \{(1,1,1)\}$.

Table 1 contains all distances relevant to computations in order to calculate $\Delta_{Max}(E)$, $\Delta_\Sigma(E)$ and $\Delta_{GMax}(E)$.

As the min in the column of $dist_{Max}$ is 1 we have $Mod(\Delta_{Max}(E)) = \{(0,1,1), (1,0,1),(1,1,0)\}$, thus the teacher has to teach two of the three languages to best satisfy the class when the criterion to solve conflicts is Δ_{Max}. Similarly as the min in the column of $dist_{\Sigma}$ is 2 we have $Mod(\Delta_{\Sigma}(E)) = \{(0,0,1),(1,0,1)\}$, thus the teacher has to teach both SQL and O_2 or O_2 alone to best fit the class when the criterion to solve conflicts is Δ_{Σ}. Finally as the min in the column of $dist_{\Delta_{GMax}}$ is $(1,1,0)$ we have $Mod(\Delta_{GMax}(E)) = \{(1,0,1)\}$, thus the teacher has to teach SQL and O_2 to best satisfy the class when the criterion to solve conflicts is Δ_{GMax}.

As we can expect the result of the merging highly depends on the operator we choose. Note in particular that the Δ_{Max} operator has selected interpretations that satisfy as much as possible each student, whereas the Δ_{Σ} operator has selected interpretations that satisfy the majority of students. Notice also that in this example the Δ_{GMax} operator selects the interpretation chosen by both Δ_{Max} and Δ_{Σ} operators, showing its good behaviour.

We will see now the logical properties of these three operators.

We first show that Δ_{Max} is not a merging operator.

Theorem 23 *Δ_{Max} satisfies postulates $(A1-A5)$, $(A7')$ and (A_{it}) but it doesn't satisfy $(A6)$.*

Proof: The proof of $(A1-A3)$ and $(A5)$ is straightforward. To prove that $(A4)$ is satisfied suppose $K \wedge K' \vdash \bot$. We consider two cases: $dist(K,K') = 1$ or $dist(K,K') > 1$ If $dist(K,K') = 1$ then $\exists I \in Mod(K), \exists J \in Mod(K')$ such that $dist(I,J) = 1$, so as $dist(I,J)$ is minimum $I \in Mod(\Delta(K \sqcup K'))$ and $J \in Mod(\Delta(K \sqcup K'))$, so $\Delta(K \sqcup K') \not\vdash K$. Otherwise $dist(K,K') > 1$, and then $\exists I \in Mod(K), \exists J \in Mod(K')$ $\forall I' \in Mod(K), \forall J' \in Mod(K')$ $dist(I,J) \leq dist(I',J')$ and $dist(I,J) > 1$. But it is easy to see that if $dist(I,J) = a > 1$ then there exists $L \in \mathcal{W}$ such that $dist(L,I) < a$ and $dist(L,J) < a$, so $dist_{Max}(L, K \sqcup K') < a$. Therefore $L <^{Max}_{K \sqcup K'} I$ so $I \notin Mod(\Delta(K \sqcup K'))$, so $\Delta(K \sqcup K') \not\vdash K$. $(A7')$ is satisfied because $\max_{\varphi \in E \sqcup K^n} dist(I,\varphi) = \max_{\varphi \in E \sqcup K} dist(I,\varphi)$. So $\Delta(E \sqcup K^n) = \Delta(E,K)$. As $(A7')$ is satisfied, $(A7)$ is satisfied. In order to show that $(A6)$ is not satisfied consider the example 22 and observe that if we take $E_1 = \{\varphi_1\}$ and $E_2 = \{\varphi_2, \varphi_3\}$, then $\Delta(E_1) \wedge \Delta(E_2) = form(\{(1,0,1)\})$ is consistent, and $\Delta(E_1 \sqcup E_2) = form(\{(0,1,1),(1,0,1),(1,1,0)\})$, so $\Delta(E_1 \sqcup E_2) \nvdash \Delta(E_1) \wedge \Delta(E_2)$.

It remains to show that (A_{it}) holds. First, by induction on $dist(K,K')$ we prove that

$$\exists n \text{ such that } \Delta^n_{Max}(K',K) \vdash K \quad (*)$$

If $dist(K,K') = 0$ the proof is straightforward. Suppose $dist(K,K') = 1$. Then $\exists I \in Mod(K)$ $\exists J \in Mod(K')$ $dist(I,J) = 1$. So $I \in \Delta_{Max}(K,K')$ and then, by (A2), $\Delta^2_{Max}(K',K) = \Delta_{Max}(\Delta_{Max}(K',K),K) = \Delta_{Max}(K',K) \wedge K$. So $\Delta^2_{Max}(K',K) \vdash K$. Suppose that $dist(K,K') > 1$. Put $a = dist(K',K)$, i.e. $\exists I \in Mod(K)$ $\exists J \in Mod(K')$ $dist(I,J) = a$. Let $a/2$ be the integer part of the quotient of a by 2. Since I and J disagree on a letters, we can find an interpretation I' such that I' agrees with I on the letters on which I and J agree, and I' agrees with J on $a/2$ letters on which I and J disagree and I' agrees with I for the $a/2$ remaining letters if a is even and for the $a/2+1$ remaining letters if a is odd. So we have $dist(I',K) \leq a/2$ and $dist(I',K') \leq a/2$ if a is even or $dist(I',K') \leq a/2+1$ if a is odd.
If a is even then $dist_{Max}(I',\{K,K'\}) \leq a/2$, so if $J' \in Mod(\Delta_{Max}(K,K'))$ then $dist_{Max}(J',\{K,K'\}) \leq a/2$. So we have that if $dist(K,K') = a$ with $a > 1$ then $dist(K, \Delta_{Max}(K,K')) \leq a/2$. By induction hypothesis there exists n such that $\Delta^n_{Max}(\Delta_{Max}(K,K'),K) \vdash K$ that is $\Delta^{n+1}_{Max}(K',K) \vdash K$. The case where a is odd is similar. Now (A_{it}) follows from $(*)$ by putting $K' = \Delta(E \sqcup K)$. ∎

The operator Δ_{Σ} is a majority merging operator as stated in the following theorem.

Theorem 24 *Δ_{Σ} satisfies postulates $(A1-A6)$, $(M7)$ and (A_{it}).*

Proof: We will prove that the assignment $E \mapsto \leq^{\Sigma}_{E}$ is a majority syncretic assignment. Then by theorem 18 we conclude that Δ_{Σ} satisfies $(A1-A6)$ and $(M7)$. Let's verify the conditions of a majority syncretic assignment:

1. If $I \in Mod(E)$ and $J \in Mod(E)$, then $dist_{\Sigma}(I,E) = 0$ and $dist_{\Sigma}(J,E) = 0$, so $I \simeq_E J$.

2. If $I \in Mod(E)$ and $J \notin Mod(E)$, then $dist_{\Sigma}(I,E) = 0$ and $dist_{\Sigma}(J,E) > 0$, so $I <_E J$.

3. Straightforward.

4. Suppose $K \wedge K' \vdash \bot$, so $dist(K,K') > 0$. So $\exists I \in Mod(K), \exists J \in Mod(K')$ $\forall I' \in Mod(K), \forall J' \in Mod(K')$ $dist(I,J) \leq dist(I',J')$ and $dist(I,J) = a > 1$. It is easy to see that $a = \min\{dist_{\Sigma}(L, K \sqcup K') : L \in \mathcal{W}\}$ thus $I \in Mod(\Delta(K \sqcup K'))$ and $J \in Mod(\Delta(K \sqcup K'))$, so $\Delta(K \sqcup K') \not\vdash K$.

5. If $I \in \min(\leq_{E_1})$ and $I \in \min(\leq_{E_2})$, then $\forall J$ $dist_{\Sigma}(I,E_1) \leq dist_{\Sigma}(J,E_1)$ and

$dist_\Sigma(I,E_2) \leq dist_\Sigma(J,E_2)$. So $\forall J\ dist_\Sigma(I,E_1) + dist_\Sigma(I,E_2) \leq dist_\Sigma(J,E_1) + dist_\Sigma(J,E_2)$. By definition of $dist_\Sigma$ is easy to see that for any L,E,E', $dist_\Sigma(L,E \sqcup E') = dist_\Sigma(L,E) + dist_\Sigma(L,E')$. Then $\forall J\ dist_\Sigma(I,E_1 \sqcup E_2) \leq dist_\Sigma(J,E_1 \sqcup E_2)$. So $I \in \min(\leq_{E_1 \sqcup E_2})$.

6. If $\min(\leq_{E_1}) \cap \min(\leq_{E_2}) \neq \emptyset$, then $\exists J$ s.t. $J \in \min(\leq_{E_1})$ and $J \in \min(\leq_{E_2})$. Suppose $I \notin \min(\leq_{E_1})$, then $dist_\Sigma(J,E_1) < dist_\Sigma(I,E_1)$ and $dist_\Sigma(J,E_2) \leq dist_\Sigma(I,E_2)$. So $dist_\Sigma(J,E_1) + dist_\Sigma(J,E_2) < dist_\Sigma(I,E_1) + dist_\Sigma(I,E_2)$. Then $dist_\Sigma(J,E_1 \sqcup E_2) < dist_\Sigma(I,E_1 \sqcup E_2)$. Then $I \notin \min(\leq_{E_1 \sqcup E_2})$.

7. We have to find a n such that $\min(\leq_{E \sqcup K^n}) \subseteq Mod(K)$. Consider $x = \max_{I \in \mathcal{W}} dist_\Sigma(I,E)$, i.e. x is the distance of the furthest interpretation from E. We choose $n = x+1$, it is easy to see that if $I \in Mod(K)$ then $dist_\Sigma(I, E \sqcup K^n) < n$. And if $I \notin Mod(K)$ then $dist_\Sigma(I,E \sqcup K^n) \geq n$. So if $I \in \min(\leq_{E \sqcup K^n})$ then $I \in Mod(K)$.

Now we prove that (A_{it}) holds. We want to show that $\exists n\ \Delta^n_\Sigma(K',K) \vdash K$. Let a be the distance between K and K'. Take $I \in Mod(K)$ and $J \in Mod(K')$ such that $dist(I,J) = a$. It is easy to see that $a = \min\{dist_\Sigma(L, K \sqcup K') : L \in \mathcal{W}\}$ thus $I \in Mod(\Delta(K \sqcup K'))$ and then $\Delta_\Sigma(\Delta_\Sigma(K' \sqcup K),K) \vdash K$. Therefore $\exists n\ \Delta^n_\Sigma(K',K) \vdash K$. And with $K' = \Delta_\Sigma(E \sqcup K)$ we have $\exists n\ \Delta^n_\Sigma(E,K) \vdash K$. ∎

Now, we will state some lemmas in order to prove that Δ_{GMax} has desirable properties.

Definition 25 *Let L_1 and L_2 be two lists of n numbers sorted in descending order. We define $L_1 \odot L_2$ the list obtained by sorting in descending order the concatenation of L_1 with L_2.*

Lemma 26 *Let L_1, L'_1, L_2, L'_2 be 4 lists of integers sorted in descending order. If $L_1 \leq_{lex} L'_1$ and $L_2 \leq_{lex} L'_2$ then $L_1 \odot L_2 \leq_{lex} L'_1 \odot L'_2$.*

Proof: Suppose that $L_1 \leq L'_1$ and $L_2 \leq L'_2$. It is easy to see that the two following inequalities hold: $L_1 \odot L_2 \leq_{lex} L'_1 \odot L_2$ and $L_2 \odot L'_1 \leq_{lex} L'_2 \odot L'_1$. So by transitivity $L_1 \odot L_2 \leq_{lex} L'_1 \odot L'_2$. ∎

Lemma 27 *Let L_1, L'_1, L_2, L'_2 be 4 lists of integers sorted in descending order. If $L_1 \leq_{lex} L'_1$ and $L_2 <_{lex} L'_2$ then $L_1 \odot L_2 <_{lex} L'_1 \odot L'_2$.*

Proof: With the assumptions it is easy to see that $L_1 \odot L_2 \leq_{lex} L'_1 \odot L_2$ and $L_2 \odot L'_1 <_{lex} L'_2 \odot L'_1$. We conclude by transitivity of $\leq_{lex}$. ∎

The operator Δ_{GMax} is a true arbitration operator as showed in the following theorem.

Theorem 28 *The operator Δ_{GMax} satisfies postulates $(A1-A6)$ and (A_{it}). Furthermore Δ_{GMax} satisfies $(A7)$ iff $card(\mathcal{P}) > 1$. But it doesn't satisfy $(A7')$.*

Proof: In order to show that $GMax$ satisfies $(A1-A7)$ we use the representation theorem and we show that the assignment $E \mapsto \leq^{Gmax}_E$ is a fair syncretic assignment.

1. If $I \in Mod(E)$ and $J \in Mod(E)$, then $\forall K_i \in E\ I \in Mod(K_i)$ and $J \in Mod(K_i)$, then $L_I = (0,\dots,0)$ and $L_J = (0,\dots,0)$, so $I \simeq_E J$.

2. If $I \in Mod(E)$ and $J \notin Mod(E)$, then $L_I = (0,\dots,0)$ and $L_J \neq (0,\dots,0)$, so $I <_E J$.

3. If $E_1 \leftrightarrow E_2$, then is obvious that $\leq_{E_1} = \leq_{E_2}$.

4. This property is proved in a similar way as (A_4) for Δ_{Max} (theorem 23).

5. If $I \in \min(\leq_{E_1})$ and $I \in \min(\leq_{E_2})$, then $\forall J \in \mathcal{W}\ L^{E_1}_I \leq_{lex} L^{E_1}_J$ and $L^{E_2}_I \leq_{lex} L^{E_2}_J$. So, by lemma 26, we have $\forall J\ L^{E_1 \sqcup E_2}_I \leq_{lex} L^{E_1 \sqcup E_2}_J$. Then $I \in \min(\leq_{E_1 \sqcup E_2})$.

6. If $\min(\leq_{E_1}) \cap \min(\leq_{E_2}) \neq \emptyset$ and $I \notin \min(\leq_{E_1})$, let $J \in \min(\leq_{E_1}) \cap \min(\leq_{E_2})$, so $L^{E_1}_J <_{lex} L^{E_1}_I$ and $L^{E_2}_J \leq_{lex} L^{E_2}_I$, and by lemma 27 follows $L^{E_1 \sqcup E_2}_J <_{lex} L^{E_1 \sqcup E_2}_I$. Then $I \notin \min(\leq_{E_1 \sqcup E_2})$.

7. Consider a knowledge base K'. We will show that if there are 2 or more propositional variables then there exists a K s.t. $K' \nvdash K$ and $\forall n\ \min(\leq_{K' \sqcup K^n}) = \min(\leq_{K' \sqcup K})$. We consider 2 cases, first if $card(Mod(K')) > 1$ then let $I \in Mod(K')$, we choose $K = form(I)$. So, by condition 1 and 2, $\min(\leq_{K' \sqcup K}) = \{I\}$ and $\forall n\ \min(\leq_{K' \sqcup K^n}) = \{I\}$. Hence $\forall n\ \min(\leq_{K' \sqcup K^n}) = \min(\leq_{K' \sqcup K})$. Second, if $card(Mod(K')) = 1$, let $Mod(K') = \{J\}$, we choose $K = form(I)$ s.t. $dist(I,J) = 2$, this is possible because there are at least two propositional variables. So there exists I' s.t. $dist(I',I) = 1$ and $dist(I',J) = 1$. So $\min(\leq_{K' \sqcup K^n}) = \{I' : dist(I',I) = 1 \text{ and } dist(I',J) = 1\}$ otherwise if $\exists J'$ such that $dist(J',K) = 0$ then $dist(J',K') \geq 2$ or if $dist(J',K') = 0$ then $dist(J',K) \geq 2$, and so $L_{I'} < L_{J'}$. So $\forall n\ \min(\leq_{K' \sqcup K^n}) = \{I' : dist(I',I) = 1 \text{ and } dist(I',J) = 1\}$. Then $\forall n\ \min(\leq_{K' \sqcup K^n}) = \min(\leq_{K' \sqcup K})$. Conversely suppose that $\mathcal{P} = \{p\}$. Put $K' = p$. Then the only consistent K (up to logical equivalence) such

Table 2: Summary Table

	A1	A2	A3	A4	A5	A6	A7	A7′	M7	A_{it}
Max	✓	✓	✓	✓	✓	–	✓	✓	–	✓
Σ	✓	✓	✓	✓	✓	✓	–	–	✓	✓
$GMax$	✓	✓	✓	✓	✓	✓	✓	–	–	✓

that $K' \not\vdash K$ is $K = \neg p$ but $\Delta_{GMax}(K' \sqcup K^n) = \neg p$ for any $n \geq 2$ whereas $\Delta_{GMax}(K' \sqcup K) = \neg p \vee p$.

To show that Δ_{GMax} doesn't satisfy $(A7')$ consider the following example: Suppose that $\mathcal{P} = \{p, q\}$ and that $K' = \neg p \wedge \neg q$ and $K = \neg p \wedge q$. It is easy to see that $\Delta_{GMax}(K \sqcup K') = \neg p$ whereas $\Delta_{GMax}(K' \sqcup K^n) = \neg p \wedge q$ for any $n \geq 2$.

Finally the proof that the postulate (A_{it}) holds for Δ_{GMax} goes exactly the same way that for Δ_{Max} (theorem 23). ∎

Actually $GMax$ operator is a refinement of the Max operator. More precisely we have the following observation the proof of which is straightforward:

Observation 29 $\Delta_{GMax}(E) \vdash \Delta_{Max}(E)$.

We end this section referring the reader to table 2 which sums up the properties of operators defined above. It is filled using the results of this section together with some results of section 3. The symbol ✓ (respectively –) in a square means that the corresponding operator satisfies (resp. does not satisfy) the corresponding postulate.

6 Conclusion and future work

We have proposed in this paper a set of postulates that a rational merging operator has to satisfy. We have made a distinction between arbitration operators striving to minimize individual dissatisfaction and majority operators striving to minimize global dissatisfaction. The fairness postulate is the key postulate in this distinction. We have shown that our characterization is equivalent to a family of pre-orders on interpretations. We show the consistency of the axiomatic characterization by giving examples of operators. In particular, we have proposed a new rational merging operator called Δ_{GMax} and shown that it is an arbitration operator.

Actually, in a committee, all the protagonists do not have the same weight on the final decision and so one needs to weight each knowledge base to reflect this. The idea behind weights is that the higher weight a knowledge base has, the more important it is. If the knowledge bases reflect the view of several people, weights could represent, for example, the cardinality of each group. We want to characterize logically the use of this weights. Majority operators are close to this idea of weighted operators since they allow to take cardinalities into account. But a more subtle treatment of weights in merging is still to do, in particular the notion of weighted arbitration operators is missing.

In this work the result of a merging is a subset of the set of all interpretations but a lot of systems have to conform to a set of integrity constraints, for that reason it is interesting to be able to merge some knowledge sets in the presence of these constraints [LMb]. And so one has to restrain the result of the merging to be a subset of the set of allowed interpretations. Suppose that these integrity constraints are denoted by the knowledge base IC. If we consider a weighted rational merging, a way to incorporate integrity constraints is to add IC to E with a weight “infinity”. Thus we would ensure that the interpretations selected were models of IC. Intuitively, it amounts to consider a person in the committee whose view is unquestionable and therefore one has to choose among the alternatives given by that person.

But the best way to include integrity constraints seems to be to select the minimal models in the models of the IC base rather than in $\mathcal{W}$. Intuitively, we restrict the choices of interpretations to those which satisfy IC. It is in a sense what Revesz called model fitting operators [Rev97].

In that paper we use only the Dalal's distance to define the distance between two interpretations, it would be interesting to study operators defined with other distances, in particular distances which give partial orders.

Notice also that the three merging operators defined in the paper are based on the Dalal's distance. But if one chooses an other distance between interpretations and keeps the same definitions, then one obtains other merging operators. So, more exactly, we have defined in this paper three families of merging operators, function of the definition of the distance between interpretations. It would be interesting to find what the minimum conditions on that distance are to ensure that the operators satisfy the axiomatic characterization.

Two other points of interest are to study merging operators which are not defined from a distance and to study syntactic definition of merging operators.

Acknowledgements

We would like to thank D. Makinson, P. Liberatore, P. Revesz and two anonymous referees for their comments which have improved the quality of this paper.

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.

[Arr63] K. J. Arrow. *Social choice and individual values*. Wiley, New York, second edition, 1963.

[BI84] A. Borgida and T. Imielinski. Decision making in committees: A framework for dealing with inconsistency and nonmonotonicity. In *Proceedings Workshop on Nonmonotonic Reasoning*, pages 21–32, 1984.

[BKM91] C. Baral, S. Kraus, and J. Minker. Combining multiple knowledge bases. *IEEE Transactions on Knowledge and Data Engineering*, 3(2):208–220, 1991.

[BKMS92] C. Baral, S. Kraus, J. Minker, and V. S. Subrahmanian. Combining knowledge bases consisting of first-order theories. *Computational Intelligence*, 8(1):45–71, 1992.

[Dal88] M. Dalal. Updates in propositional databases. Technical report, Rutgers University, 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.

[Lin96] J. Lin. Integration of weighted knowledge bases. *Artificial Intelligence*, 83(2):363–378, 1996.

[LMa] J. Lin and A. O. Mendelzon. Knowledge base merging by majority. Manuscript.

[LMb] J. Lin and A. O. Mendelzon. Merging databases under constraints. To appear in International Journal of Cooperative Information System.

[LS95] P. Liberatore and M. Schaerf. Arbitration: A commutative operator for belief revision. In *Proceedings of the Second World Conference on the Fundamentals of Artificial Intelligence*, pages 217–228, 1995.

[LS98] P. Liberatore and M. Schaerf. Arbitration (or how to merge knowledge bases). *IEEE Transactions on Knowledge and Data Engineering*, 10(1), 1998. To appear.

[Rev93] P. Z. Revesz. On the semantics of theory change: arbitration between old and new information. In *Proceedings of the* 12th *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.

[Sav71] L. J. Savage. *The foundations of statistics*. Dover Publications, New York, 1971. Second revised edition.

[Sub94] V. S. Subrahmanian. Amalgamating knowledge bases. *ACM Transactions on Database systems.*, 19(2):291–331, 1994.

Characterizing the Semantics of Terminological Cycles in $\mathcal{ALN}$ Using Finite Automata

Ralf Küsters
LuFg Theoretische Informatik, RWTH Aachen, Ahornstr. 55, 52074 Aachen, Germany
e-mail: kuesters@cantor.informatik.rwth-aachen.de

Abstract

The representation of terminological knowledge may naturally lead to terminological cycles. In addition to descriptive semantics, the meaning of cyclic terminologies can also be captured by fixed-point semantics, namely, greatest and least fixed-point semantics. To gain a more profound understanding of these semantics and to obtain inference algorithms as well as complexity results for inconsistency, subsumption, and related inference tasks, this paper provides automata theoretic characterizations of these semantics. More precisely, the already existing results for $\mathcal{FL}_0$ are extended to the language $\mathcal{ALN}$, which additionally allows for primitive negation and number-restrictions. Unlike $\mathcal{FL}_0$, the language $\mathcal{ALN}$ allows to express inconsistent concepts, which makes non-trivial extensions of the characterizations and algorithms necessary. Nevertheless, the complexity of reasoning does not increase when going from $\mathcal{FL}_0$ to $\mathcal{ALN}$. This distinguishes $\mathcal{ALN}$ from the very expressive languages with fixed-point operators proposed in the literature. It will be shown, however, that cyclic $\mathcal{ALN}$-terminologies are expressive enough to capture schemas in certain semantic data models.

1 INTRODUCTION

Cyclic terminologies were first investigated by B. Nebel (Nebel, 1987; Nebel, 1990a; Nebel, 1990b), who has introduced fixed-point semantics (namely, greatest and least fixed-point semantics) in addition to descriptive semantics to capture the meaning of such terminologies. Which of the semantics should be preferred depends on the particular representation problem at hand. Furthermore, Nebel has shown—by proving the finite model property—that for the language $\mathcal{ALN}$, which allows for concept conjunction, (universal) value-restriction, primitive negation, and unqualified number-restrictions, subsumption w.r.t. cyclic terminologies and the descriptive semantics is decidable.

In order to gain a more profound understanding of all three semantics as well as more feasible decision algorithm and complexity results for subsumption, F. Baader (Baader, 1996b) has proposed automata theoretic characterizations of the three semantics for the small representation language $\mathcal{FL}_0$, which allows for concept conjunction and (universal) value-restrictions. Following this approach, B. Nebel (Nebel, 1991) has given an automata theoretic characterization of equivalence of concepts w.r.t. cyclic $\mathcal{FL}_0$-terminologies. Although (Nebel, 1991) introduces the semantics for $\mathcal{ALN}$, the technical results are restricted to the sublanguage $\mathcal{FL}_0$.

Since $\mathcal{FL}_0$ is not expressive enough for most practical representation problems, in this paper the results for $\mathcal{FL}_0$ are extended to the language $\mathcal{ALN}$. Generalizing the results from $\mathcal{FL}_0$ to $\mathcal{ALN}$ is not trivial due to inconsistent concepts, which are expressible in $\mathcal{ALN}$, but not in $\mathcal{FL}_0$. Technically, this fact is dealt with by introducing the notion of so-called *excluding words*. The new complexity results presented in this paper are summarized in Table 1 for all three semantics both for general (i.e., possibly cyclic) and weak-acyclic terminologies (allowing only for ε-cycles) where 'PSp' denotes PSPACE-completeness and 'NP' NP-completeness. The complexity upper bounds are proved by deriving decision algorithms from the automata theoretic characterizations of inconsistency and subsumption presented in this work. The complexity lower bounds are (more or less straightforward) consequences of known complexity results for $\mathcal{FL}_0$-terminologies and schemas.

Table 1: Complexity Results for $\mathcal{ALN}$-Terminologies

	descriptive	gfp	lfp
Subsumption general	PSp	PSp	PSp
Inconsistency general	PSp	PSp	PSp
Inconsistency (weak-)acyclic	NP	NP	NP

Terminological cycles in much more expressive extensions of $\mathcal{FL}_0$ have been investigated in (Schild, 1994) and (Giacomo and Lenzerini, 1994). K. Schild has extended the language $\mathcal{ALC}$ by the fixed-point operators of the μ-calculus to $\mu\mathcal{ALC}$, and has shown—among other results—that $\mu\mathcal{ALC}$ is more expressive than general $\mathcal{ALC}$-terminologies.[1] Moreover, the language $\mu\mathcal{ALC}$ has been extended in (Giacomo and Lenzerini, 1994) by (qualified) number-restrictions to $\mathcal{ALCQ}\mu$. Thus, the language $\mathcal{ALCQ}\mu$ contains $\mathcal{ALN}$. Consistency as well as subsumption for $\mathcal{ALCQ}\mu$-concepts is EXPTIME-complete, whereas these problems are merely PSPACE-complete for $\mathcal{ALN}$ (see Table 1). This justifies to consider the restricted case separately since it is likely that implemented decision algorithms for PSPACE-complete problems are easier to optimize than those for EXPTIME-complete problems (Giunchiglia and Sebastiani, 1996). For both $\mathcal{ALCQ}\mu$ and $\mathcal{ALN}$, the important inference problems can be decided with the help of finite automata. However, the automata for $\mathcal{ALCQ}\mu$ are of exponential size and they are tree automata reflecting certain semantic structures, while the automata for $\mathcal{ALN}$ are finite automata on words that are merely syntactic variants of $\mathcal{ALN}$-terminologies.

Cycles can increase the complexity of inference problems or may even lead to undecidability. For instance, subsumption for general $\mathcal{FL}_0$-terminologies which allow for feature agreements is undecidable (Nebel, 1991), whereas this problem is decidable for acyclic terminologies.[2] In this paper, we exhibit another instance of this phenomenon: It is shown that inconsistency w.r.t. acyclic $\mathcal{ALN}$-terminologies is NP-complete, whereas it is PSPACE-complete for general $\mathcal{ALN}$-terminologies (see Table 1).

Nevertheless, the literature points out the need for cyclic definitions of concepts (see, e.g., (MacGregor, 1992; Buchheit et al., 1994; Calvanese, 1996)). For instance, description logics can be used to describe schemas in many semantic and object-oriented data models (see, e.g., (Borgida, 1995; Bergamaschi and Sartori, 1992; Calvanese et al., 1994)). These schemas are often cyclic; e.g., the Boss of an Employee is a Manager who is himself an Employee. Unlike terminologies, which consist of concept definitions, schemas state only necessary (rather than necessary and sufficient) conditions for concepts. In this paper, however, it is shown that the important inference problems for so-called (cyclic) $\mathcal{SL}_{dis}$-schemas, which where introduced in (Buchheit et al., 1994), can be reduced to corresponding problems for (cyclic) $\mathcal{ALN}$-terminologies. Hence, the decision algorithms for terminologies presented below can be used to decide inference problems for schemas. This yields new proofs for the complexity upper bounds for such schemas, and moreover, for a more expressive schema language since it allows for arbitrary number-restrictions. Conversely, existing results for complexity lower bounds of schemas can be used to derive complexity results for terminologies.

In the next section we introduce some basic definitions and state simple properties. In section 3 we characterize the three semantics. The subsequent sections contain characterizations of inconsistency as well as subsumption for all three semantics. Within these sections the results for the greatest fixed-point semantics are presented in detail; the extensions required by the descriptive and least fixed-point semantics are briefly described.

2 BASIC DEFINITIONS

The basic syntactic elements of a concept language are concepts and roles. Let A denote an atomic concept, i.e., a concept name, as well as C, D arbitrary concepts, R a role name, and n a non-negative integer. Then *$\mathcal{ALN}$-concepts* are formed by means of the following syntax rules:

$$C, D \longrightarrow A \mid \neg A \mid C \sqcap D \mid \forall R.C \mid (\geq n\, R) \mid (\leq n\, R)$$

A *(general) $\mathcal{ALN}$-terminology* T consists of a finite set of concept definitions $A = C$ where A denotes an atomic concept and C an $\mathcal{ALN}$-concept. In addition, it is required that for every atomic concept A in T there is at most one concept definition with left-hand side A. We call atomic concepts appearing on the left-hand side of some concept definition in T *defined*, otherwise *primitive*. For an $\mathcal{ALN}$-terminology T only primitive negation is allowed, i.e., if $\neg A$ is a sub-concept in T, then A must be primitive.

A terminology is *cyclic* if there exists at least one

[1]To ensure the existence of least and greatest fixed-point models, recursively defined concepts must occur positively in their definition.

[2]One can use the technique described in (Aït-Kaci, 1984) to obtain this result (see also (Hollunder and Nutt, 1990)).

atomic concept which (directly or indirectly) occurs in its own definition. Formally, cycles are defined as follows: Let A denote a defined concept and B an atomic concept. We say that A *directly uses* B if B occurs on the right-hand side of the concept definition $A = C$ in T. Now let *uses* denote the transitive closure of *directly uses*. Then a terminology T is cyclic if there is a defined concept in T that *uses* itself, otherwise T is called *acyclic*.

The following is an example (taken from (Nebel, 1991)) of a cyclic $\mathcal{ALN}$-terminology consisting of only one concept definition:

$$\begin{aligned} T: \mathsf{Human} \;=\; & \mathsf{Mammal} \sqcap (\geq 2\ \mathsf{parents}) \sqcap \\ & (\leq 2\ \mathsf{parents}) \sqcap \forall \mathsf{parents}.\mathsf{Human} \end{aligned}$$

Here, Human is a defined concept, Mammal a primitive concept, and parents a role. Intuitively, this terminology defines human beings as those mammals having exactly two parents all of whom are human beings.

In order to define the semantics of $\mathcal{ALN}$-terminologies formally, we—similarly to the first-order predicate logic—have to introduce the notion "interpretation".

An *interpretation* I consists of a domain $dom(I)$ and a mapping assigning a subset C^I of $dom(I)$ (the *extension of* C) to every atomic concept C as well as a binary relation R^I over $dom(I)$ (the *extension of* R) to every atomic role R. This interpretation is extended to $\mathcal{ALN}$-concepts as defined in Table 2 where $R^I(d) := \{e \in dom(I) \mid (d,e) \in R^I\}$ denotes the set of R-*successors* of d in I. For convenience we write

Table 2: Semantics of $\mathcal{ALN}$-Concepts

Syntax	Semantics
$\neg A$	$dom(I) \setminus A^I$
$C \sqcap D$	$C^I \cap D^I$
$\forall R.C$	$\{d \in dom(I); R^I(d) \subseteq C^I\}$
$(\leq n\ R)$	$\{d \in dom(I); \|R^I(d)\| \leq n\}$
$(\geq n\ R)$	$\{d \in dom(I); \|R^I(d)\| \geq n\}$

$\forall W.C$ instead of $\forall R_1.\forall R_2 \cdots \forall R_n.C$, $n \geq 0$, where W denotes the word $R_1 \cdots R_n$. According to this abbreviation, let W^I be the composition $R_1^I \circ \cdots \circ R_n^I$ of the relations R_i^I, $1 \leq i \leq n$. For the empty word, ε^I denotes the identity relation. As for an atomic role, $W^I(d)$ denotes the set of W-successors of d in I.

An interpretation I is a *model* of a terminology T iff $A^I = C^I$ for all concept definitions $A = C$ in T. The *descriptive semantics* of T is defined by the set of all models I of T.

In presence of cyclic dependencies the descriptive semantics does not capture the intuitive meaning of a terminology in any case. Therefore, the meaning of cyclic terminologies is also specified by so-called fixed-point semantics, which have been introduced by B. Nebel (Nebel, 1991). In order to define these semantics, we divide the interpretation I of the terminology T into the *primitive interpretation* J and the interpretation of the defined concepts of T (Human in the example). The primitive interpretation determines the extensions of the primitive concepts and roles in T (Mammal and role in the example). For acyclic terminologies the primitive interpretation J of T can uniquely be extended to a model I of T, i.e., the extensions of the defined concepts in T are uniquely determined by J and T. However, in the presence of cyclic dependencies there may exist several possible extensions of the primitive interpretation J to a model I of T. In such a model the extensions of the defined concepts are fixed-points of the following mapping:

Definition 1.
Let T be a terminology consisting of the concept definitions $A_1 = D_1, \ldots, A_n = D_n$, and let J be a primitive interpretation. The mapping $T_J : (2^{dom(J)})^n \longrightarrow (2^{dom(J)})^n$ is defined as follows: Let $\underline{A} \in (2^{dom(J)})^n$ be a tuple, namely, a tuple of extensions of the defined concepts, and I the interpretation given by J and $\underline{A}$. Then $T_J(\underline{A}) := (D_1^I, \ldots, D_n^I)$. ◇

Note, that for $\mathcal{ALN}$ the mapping T_J is monotonic where the tuples are ordered componentwise by set inclusion. Thus, for every $\mathcal{ALN}$-terminology T and every primitive interpretation J there is always a unique greatest and least fixed-point of T_J. Furthermore, it is not hard to see that if I is a model of T with primitive interpretation J, then the tuple of the extensions of the defined concepts in T is a fixed-point of T_J, i.e., I is specified by J and a fixed-point of T_J.

Now, we are equipped to define the fixed-point semantics mentioned above. The *lfp-semantics* allows only those models of T as admissible models which come from a primitive interpretation J and the least fixed-point of the mapping T_J (*lfp-models*); analogously, the *gfp-semantics* allows only those models of T as admissible models which come from a primitive interpretation J and the greatest fixed-point of the mapping T_J (*gfp-models*). For more information on these semantics see, e.g., (Nebel, 1991; Baader, 1996b; Baader, 1990; Küsters, 1997).

Let A, B be atomic concepts in the terminology T. The concept A is *inconsistent* w.r.t. T (T-inconsistent) and w.r.t. the descriptive semantics (lfp-, gfp-semantics) iff in every model (lfp-, gfp-model) of T the extension of A is empty. The concept A is *subsumed* by B w.r.t. T and the descriptive semantics (lfp-, gfp-semantics) iff for all models (lfp-, gfp-models) I of T we have: $A^I \subseteq B^I$. In this case, we write $A \sqsubseteq_T B$ ($A \sqsubseteq_{lfp,T} B$, $A \sqsubseteq_{gfp,T} B$).

Since $\mathcal{ALN}$ allows for number-restrictions, primitive negation can be dispensed with using the technique proposed in (Baader, 1996a): The concepts $\neg P$ and P in a terminology can be replaced by $(\leq 0\ R_P)$ and $(\geq 1\ R_P)$, respectively, for a new role name R_P. By this substitution one can reduce inconsistency and subsumption of concepts for $\mathcal{ALN}$-terminologies to $\mathcal{FLN}$-terminologies, that do not allow for primitive negation. Without loss of generality, we therefore restrict our attention to $\mathcal{FLN}$ in the sequel.

In order to characterize the three semantics we first have to normalize the terminology T and then associate a semi-automaton to the normalized terminology: T is called *normalized* if every right-hand side of a concept definition is a finite conjunction of concepts of the form $\forall W.C$ where W is a finite word over the role names in T and C is an atomic concept or a number-restriction. A *semi-automaton* is defined by a finite alphabet Σ, a finite set of states Q, and a finite set $E \subseteq Q \times \Sigma^* \times Q$ of *word-transitions.*

Now, the (non-deterministic) semi-automaton $\mathcal{A}_T = (\Sigma, Q, E)$ of the *normalized* $\mathcal{FLN}$-terminology T is defined as follows: The alphabet Σ of $\mathcal{A}_T$ consists of the role names occurring in T; the atomic concepts and number-restrictions of T are the states $q \in Q$ of $\mathcal{A}_T$; a concept definition $A = \forall W_1.A_1 \sqcap \cdots \sqcap \forall W_k.A_k$ gives rise to k word-transitions, where the transition from A to A_i is labeled with the word W_i (for more details see (Baader, 1996b; Küsters, 1997; Nebel, 1990b)). Note that in a semi-automaton, word-transitions can easily be eliminated by replacing each of these transitions by a sequence of new introduced transitions (labeled with letters) using new states. Therefore, if needed, $\mathcal{A}_T$ is (w.l.o.g.) supposed to be a *semi-automaton without word-transitions*, i.e., $E \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$. For the terminology of the example we obtain the semi-automaton $\mathcal{A}_T$ shown in Figure 1 where M = Mammal, H = Human, and p = parents.

Let A be an atomic concept and C an atomic concept or a number-restriction. Then $L_{\mathcal{A}_T}(A, C)$ (resp. $L(A, C)$ if the relationship to $\mathcal{A}_T$ is clear from the context) denotes the set of words (over the role names in T) that are labels of paths from A to C in $\mathcal{A}_T$. We also have to consider infinite words, i.e., elements of Σ^ω. The set $U_{\mathcal{A}_T}(A)$ (resp. $U(A)$) denotes the set of (finite and infinite) words which are labels of infinite paths in $\mathcal{A}_T$ starting from A. Note that since $\mathcal{A}_T$ allows for ε-transitions, $U(A)$ may contain finite words. For $L \subseteq \Sigma^*$ and $R \in \Sigma$ the set $L \cdot R$ is defined as $\{W \cdot R \mid W \in L\}$ where "$\cdot$" denotes concatenation of words; furthermore $L \cdot R^{-1} := \{W \in \Sigma^* \mid W \cdot R \in L\}$.

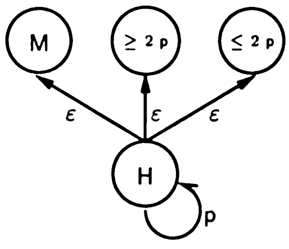


Figure 1: The Semi-Automaton $\mathcal{A}_T$

In order to characterize inconsistency and subsumption, the notion of "*requiring*" is very useful. This notion is quite similar to the notion "active chain of roles" introduced in (Calvanese, 1996). Due to number-restrictions, atomic concepts can "require" chains of role successors which have to start from every instance of such a concept.

Definition 2 (require).
Let T be an $\mathcal{FLN}$-terminology, $\mathcal{A}_T$ the corresponding semi-automaton, and A an atomic concept in T. Furthermore, let $W = R_1 \cdots R_n$ denote a finite word and $V = R_1 \cdots R_m$ a prefix of W, i.e., $m \leq n$. The word W is *required by A starting from V* iff for all i, $m \leq i < n$, there are numbers $k_{i+1} \geq 1$ such that $VR_{m+1} \cdots R_i \in L(A, (\geq k_{i+1}\ R_{i+1}))$. For infinite words this notion is defined analogously. If $V = \varepsilon$ we say "W is required by A" instead of "W is required by A starting from ε". ◇

It is easy to see that if an instance of A in definition 2 has a V-successor w.r.t. a model I of T, then it must also have a W-successor. Using this observation it follows that every instance of Human in the example has an infinite chain of ancestors: Because of parents$^j \in L$(Human, $(\geq 2$ parents)) for all $j \geq 0$, every word parents^j is required by Human.

In the following, we need two more definitions. For a semi-automaton $\mathcal{A}_T$, a set F of $\mathcal{A}_T$-states, and a role R, we define the sets:

$$\begin{aligned} \varepsilon\text{-}closure(F) &:= \{q' \mid \text{there is a state } q \in F \text{ and a (possibly empty) path from } q \text{ to } q' \text{ in } \mathcal{A}_T \text{ labeled with } \varepsilon\}, \\ next_\varepsilon(F, R) &:= \varepsilon\text{-}closure(\{q' \mid \text{there is a state } q \in \varepsilon\text{-}closure(F) \text{ and a transition from } q \text{ to } q' \text{ with label } R \text{ in } \mathcal{A}_T\}). \end{aligned}$$

Before characterizing inconsistency and subsumption, a closer look at the semantics themselves is needed.

3 CHARACTERIZING THE SEMANTICS

The automata theoretic characterizations of the three semantics for $\mathcal{FLN}$ are easy extensions of the results for $\mathcal{FL}_0$. We just have to add conditions for number-restrictions which are analogous to the conditions for primitive concepts. The characterization for the gfp-semantics is as follows:

Theorem 3 (gfp-semantics).
Let T be an $\mathcal{FLN}$-terminology, $\mathcal{A}_T$ the corresponding semi-automaton, I a gfp-model of T, and A an atomic concept occurring in T. For every $d \in dom(I)$ we have $d \in A^I$ iff the following property holds:

> For all C that are either a primitive concept or a number-restriction, and for all words $W \in L_{\mathcal{A}_T}(A, C)$ it holds that $d \in (\forall W.C)^I$. □

For lfp- and descriptive semantics an additional condition must be satisfied, which, however, coincides with the one for $\mathcal{FL}_0$ (Baader, 1996b).

Descriptive semantics: For the descriptive semantics, relationships between defined concepts also have to be taken into account. In order to characterize the descriptive semantics formally, we must introduce some more notions: As already mentioned, for a model I of T (with primitive interpretation J) the tuple of extensions of the defined concepts is a fixed-point of T_J. We can describe this fixed-point as greatest fixed-point $\underline{A}$-$gfp(T_J)$ of T_J that is less or equal a tuple $\underline{A}$. Since T_J is monotonic, it can be shown that such a fixed-point exists if $T_J(\underline{A}) \subseteq \underline{A}$ (see (Baader, 1996b; Küsters, 1997) for details). Using this observation, it is easy to see that for every model I with primitive interpretation J there is a tuple $\underline{A}$ such that $\underline{A}$-$gfp(T_J)$ exists and I consists of J and $\underline{A}$-$gfp(T_J)$. We refer to the extension of a defined concept B in $\underline{A}$ by $(\underline{A})_{index(B)}$. Now let I be a model of T defined by J and $\underline{A}$-$gfp(T_J)$. With this notation, Theorem 3 is extended by the following condition, which additionally must be satisfied (Küsters, 1997):

> For all defined concepts B and all $W \in L(A, B)$ it holds $W^I(d) \subseteq (\underline{A})_{index(B)}$.

Lfp-semantics: For the lfp-semantics there is an additional condition to be satisfied as well. This condition intuitively prohibits infinite role chains starting from $d \in A^I$. Formally, Theorem 3 is extended by the following condition:

> For all infinite paths of the form $A, W_1, C_1, W_2, C_2, \ldots$ and all individuals $d_1, d_2, d_3, \ldots \in dom(I)$ there is an $n \geq 1$ such that $(d_{n-1}, d_n) \notin W_n^I$ where $d_0 = d$.

The characterizations provide a more intuitive way to understand and to compare the different semantics. This can make the decision easier which of the semantics to prefer in a given representation task. Furthermore, we can employ these descriptions to characterize and decide inconsistency and subsumption.

4 CHARACTERIZING INCONSISTENCY

In $\mathcal{FL}_0$, inconsistent concepts only occur for lfp-semantics. Due to conflicting number restrictions, inconsistent $\mathcal{FLN}$-concepts may also occur for the gfp- and descriptive semantics.

Before formulating the characterization of inconsistency, we introduce the notion "*exclusion set*", which will become important in the construction of decision procedures both for inconsistency and for subsumption. Intuitively, an exclusion set describes a set of $\mathcal{A}_T$-states that require words leading to conflicting number-restrictions.

Definition 4 (exclusion set).
Let T be an $\mathcal{FLN}$-terminology and $\mathcal{A}_T = (\Sigma, Q, E)$ the corresponding semi-automaton without word-transitions. The set $F_0 \subseteq Q$ is called *exclusion set* w.r.t. $\mathcal{A}_T$ iff there is a word $R_1 \cdots R_n \in \Sigma^*$ and, for all i, $1 \leq i \leq n$, numbers $m_i \geq 1$, and $l > r$ such that for $F_i := next_\varepsilon(F_{i-1}, R_i)$, $1 \leq i \leq n$, the following holds: $(\geq m_i\, R_i) \in F_{i-1}$ for all $1 \leq i \leq n$ and $(\geq l\, R), (\leq r\, R) \in F_n$. ◇

Now we are ready to formulate

Theorem 5 (inconsistency).
Let T be an $\mathcal{FLN}$-terminology, $\mathcal{A}_T$ the corresponding semi-automaton (without word-transitions), and A an atomic concept in T. Then the following statements are equivalent:

1. A is T-inconsistent w.r.t. gfp-semantics.
2. There is a word $W \in \Sigma^*$ that is required by A, and there are conflicting number-restrictions $(\geq l\, R)$ and $(\leq r\, R)$, $l > r$, with $W \in L(A, (\geq l\, R)) \cap L(A, (\leq r\, R))$.
3. ε-$closure(\{A\})$ is an exclusion set. □

Proof sketch of Theorem 5 (for details see (Küsters, 1997)): It is easy to see that statement 2. implies inconsistency of A. Furthermore, proving equivalence of 2. and 3. is straightforward. The non-trivial part of the proof is to show that statement 2. is necessary for inconsistency. We show that, if statement 2. does not hold, then it is possible to define a canonical gfp-model I of T in which the extension of A is non-empty. More precise, I is a finite branching tree-model of T and the root d_0 of this tree is contained in A^I. In order to construct the primitive interpretation J of I, one first satisfies the $\geq$-restrictions required by A. Then, if 2. does not hold, it can be shown that the $\leq$-restrictions stated for A are not violated. To satisfy the $\geq$-restrictions required by A, we enumerate the (finite) words required by A, where the enumeration is ordered by the length of the words (ε, parents, parents parents, ... in the example). Starting from d_0 the tree is inductively extended according to the required words of the enumeration (in the example this leads to a binary tree since for every individual two parents are required by Human). Formally, the canonical model is defined as follows:

Definition 6 (canonical gfp-model).
Let T be an $\mathcal{FLN}$-terminology, $\mathcal{A}_T = (\Sigma, Q, E)$ the corresponding semi-automaton, and A an atomic concept in T. The primitive canonical interpretation $J = J(A, d_0)$ for A and the individual d_0 is defined as follows:

Let $W_0, W_1, W_2, W_3, \ldots$ be the infinite (w.o.l.g.; the finite case can be dealt with analogously) enumeration of the words required by A such that $W_i \neq W_j$ for all $i \neq j$, and $i < j$, implies $|W_i| \leq |W_j|$. Note that if one word is required by A, then all prefixes of this word are also required. Consequently, $W_0 = \varepsilon$ and $|W_i| < |W_{i+1}|$ implies $|W_{i+1}| = |W_i| + 1$. We define J inductively:

J_0: $dom(J_0) := \{d_0\}$; $R^{J_0} := \emptyset$ for all roles R in T; the extensions of the primitive concepts are not defined here, they will be defined for J.

J_{i+1}: For W_{i+1} there is exactly one $j < i+1$ and one role R in T such that $W_{i+1} = W_j R$. Let $m \geq 1$ be maximal such that $W_j \in L(A, (\geq m\, R))$, i.e., there is no $m' > m$ with $W_j \in L(A, (\geq m'\, R))$ (m exists since W_{i+1} is required by A, and T contains only a finite number of number-restrictions). Let $K_{i+1} := \{d_1, \ldots, d_r\} := W_j^{J_i}(d_0)$ (where $r \geq 0$) be the set of W_j-successors of d_0 in J_i. Furthermore, let $e_1^1, \ldots, e_m^1, \ldots, e_1^r, \ldots, e_m^r$ be $(r \cdot m)$ new individuals. We extend J_i to J_{i+1} by

$dom(J_{i+1}) := dom(J_i) \dot\cup \{e_1^1, \ldots, e_m^1, \ldots, e_1^r, \ldots, e_m^r\}$; $S^{J_{i+1}} := S^{J_i}$ for all roles $S \neq R$ and $R^{J_{i+1}} := R^{J_i} \dot\cup \{(d_k, e_l^k) \mid 1 \leq k \leq r,\ 1 \leq l \leq m\}$.

Now, J is defined as follows:

$dom(J) := \bigcup_{i \geq 0} dom(J_i)$; $S^J := \bigcup_{i \geq 0} S^{J_i}$ for all roles S in T; for all primitive concepts P and individuals $d \in dom(J)$ let: $d \in P^J$ iff there exists a word $W \in \Sigma^*$ such that $W \in L(A, P)$ and $d \in W^J(d_0)$.

The *canonical gfp-model* $I = I(A, d_0)$ for A and d_0 is defined as the gfp-model of $J(A, d_0)$ and T, i.e., I is defined by the primitive interpretation $J(A, d_0)$ and the greatest fixed-point of $T_{J(A,d_0)}$. ◇

Now, the following Propositions (Küsters, 1997) completes the proof of Theorem 5.

Proposition 7.
If statement 2. in Theorem 5 does not hold, then $d_0 \in A^I$ where $I = I(A, d_0)$. □

In general, for a given set of states it is a non-trivial problem to decide whether a given set of states is an exclusion set. The complexity of this problem is due to the fact that role successors can be required. It is not hard to specify a PSPACE-algorithm deciding the property of being an exclusion set. Thus, by Theorem 5 inconsistency is decidable using polynomial space (see (Küsters, 1997) for details). In (Calvanese, 1996), it has been shown that consistency of concepts w.r.t. $\mathcal{AL}$-schemas (see section 6) and descriptive semantics is PSPACE-complete. Using Theorem 17 this shows

Proposition 8.
Inconsistency in general $\mathcal{ALN}$-terminologies with respect to gfp-semantics is PSPACE-complete. □

Descriptive semantics: Every gfp-model of a terminology is also a model with respect to the descriptive semantics. Furthermore, the extensions of defined concepts w.r.t. gfp-semantics are maximal. Thus, an atomic concept A is inconsistent w.r.t. the gfp-semantics iff it is inconsistent w.r.t. the descriptive semantics. Consequently, Theorem 5 also holds for the descriptive semantics and inconsistency for the descriptive semantics is PSPACE-complete.

Lfp-semantics: Theorem 5 does not hold for the lfp-semantics due to the effect of infinite role chains. For lfp-semantics, inconsistency of A is equivalent to the disjunction of statement 2. in Theorem 5 and the following condition:

> There is a (finite or infinite) word W required by A in $U(A)$.

Definition 4 has to be adjusted accordingly, i.e., additionally to what is stated in this definition, F_0 is also an exclusion set if there is an *infinite* chain of sets $F_i := next_\varepsilon(F_{i-1}, R_i)$, $R_i \in \Sigma$, and numbers $m_i \geq 1$, $i \geq 1$, such that $(\geq m_i\ R_i) \in F_{i-1}$ for all $i \geq 1$, or if there is a *finite* chain $F_0, \ldots, F_n$ of sets satisfying analogous conditions where, in addition, F_n contains an atomic concept C in an ε-cycle, i.e., there is a non-empty path in $\mathcal{A}_T$ from C to C labeled with ε.

As for the other two semantics, one can construct a PSPACE-algorithm that tests whether a set of states is an exclusion set. Thus, inconsistency w.r.t. lfp-semantics is decidable using polynomial space (see (Küsters, 1997) for details). Unlike gfp-semantics, PSPACE-completeness of inconsistency for lfp-semantics is not an immediate consequence of PSPACE-completeness of inconsistency of concepts in $\mathcal{AL}$-schemas w.r.t. descriptive semantics (Calvanese, 1996). The proof of this complexity result for $\mathcal{AL}$-schemas is by reducing the validity of an (arbitrary) quantified Boolean formulae q to consistency of an atomic concept C_q w.r.t. a cyclic $\mathcal{AL}$-schema S_q. The problem is that S_q may require infinite chains starting from C_q although q is valid, i.e., C_q may be inconsistent w.r.t. lfp-semantics although q is valid. Fortunately, one can avoid such infinite chains in S_q such that C_q is consistent in S_q w.r.t. descriptive semantics (or even gfp-semantics) iff C_q is consistent in S_q w.r.t. lfp-semantics. This provides us with PSPACE-completeness of inconsistency in general $\mathcal{ALN}$-terminologies w.r.t. lfp-semantics (see (Küsters, 1997) for details).

Finally, we consider weak-acyclic[3] $\mathcal{ALN}$-terminologies and sketch the proof of

Proposition 9.
Consistency for concepts in *weak-acyclic* $\mathcal{ALN}$-terminologies is NP-complete with respect to the gfp-, lfp-, and descriptive semantics. □

We only consider gfp- and descriptive semantics. If T is a weak-acyclic $\mathcal{ALN}$-terminology, then the property of being an exclusion set is decidable by an NP-algorithm since for the sets F_i as defined in Definition 4 one can show that $F_{n+1} = \emptyset$, where n denotes the number of states in $\mathcal{A}_T$. From the proof in (Buchheit et al., 1994), which verifies NP-completeness of *validity* of $\mathcal{SL}_{dis}$-schemas, it follows that *consistency* of concepts with respect to $\mathcal{SL}_{dis}$-schemas is NP-complete as well. Now, since consistency w.r.t. $\mathcal{SL}_{dis}$-schemas can be reduced to consistency w.r.t. $\mathcal{ALN}$-terminologies (Theorem 17), this provides us with the complexity lower bounds stated in the above Proposition (see (Küsters, 1997) for details).

[3]A terminology T is *weak-acyclic* if $W \notin L_{\mathcal{A}_T}(A, A)$ for all *non-empty* finite words W and all atomic concepts A of T.

5 CHARACTERIZING SUBSUMPTION

For the language $\mathcal{FL}_0$, subsumption has been characterized in (Baader, 1996b) via inclusion of regular languages. With respect to gfp-semantics this characterization is as follows:

$$A \sqsubseteq_{gfp,T} B \quad \text{iff} \quad L_{\mathcal{A}_T}(B,P) \subseteq L_{\mathcal{A}_T}(A,P) \text{ for all primitive concepts } P \text{ in } T. \tag{1}$$

Due to conflicting number restrictions, the right-hand side of this equivalence is not necessary for A to be subsumed by B w.r.t. $\mathcal{FLN}$-terminologies: The atomic concept A may be "excluded" by a word that is contained in $L_{\mathcal{A}_T}(B,P)$, but not in $L_{\mathcal{A}_T}(A,P)$. Intuitively, the words W excluding A are exactly those words such that $A \sqsubseteq_{gfp,T} (\forall W.\bot)$.[4] To simplify the formal definition of the notion "exclusion" we may (without loss of generality) assume that an $\mathcal{FLN}$-terminology contains no $\leq$-restrictions of the form $(\leq 0\ R)$ since such a term can be substituted by $\forall R.\bot$. In the following we consider only terminologies without $(\leq 0\ R)$ and call them $\mathcal{FLN}^r$-terminologies.

Definition 10 (exclusion).
Let T be an $\mathcal{FLN}^r$-terminology, $\mathcal{A}_T = (\Sigma, Q, E)$ the corresponding semi-automaton, and A an atomic concept in T. The word $W \in \Sigma^*$ *excludes* A iff there exist a prefix $V \in \Sigma^*$ of W, a word $V' \in \Sigma^*$ as well as conflicting number-restrictions $(\geq l\ R)$, $(\leq r\ R)$, $l > r$, such that $VV' \in L(A, (\geq l\ R)) \cap L(A, (\leq r\ R))$ and VV' is required by A starting from V. Let E_A denote the set of words excluding A (*A-excluding words*). ◇

It is easy to see that if a word W excludes a concept A, every individual that has a W-successor cannot be an instance of A, i.e., $A \sqsubseteq_{gfp,T} (\forall W.\bot)$. In the following example, the concept A is excluded by the word RS, and we will formally show that no instance of A has RS-successors:

Let T be the $\mathcal{FLN}^r$-terminology with corresponding automaton $\mathcal{A}_T$ given in Figure 2. Furthermore, let I be a gfp-model of T and d an individual with $d \in A^I$. Assume $e \in (RS)^I(d)$. Since $RS \in L_{\mathcal{A}_T}(A, (\geq 1\ Q))$, there must be a Q-successor f of e. Further, since

[4]The symbol $\bot$ denotes the empty concept, which is equivalent to, e.g., $(\geq 2\ R) \sqcap (\leq 1\ R)$.

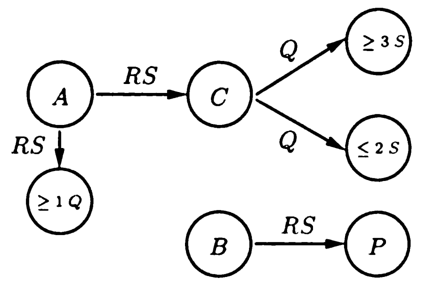


Figure 2: A Terminology Defined by Its Automaton

$RSQ \in L_{\mathcal{A}_T}(A, (\geq 3\ S)) \cap L_{\mathcal{A}_T}(A, (\leq 2\ S))$, the characterization of the gfp-semantics yields the contradiction: $f \in (\geq 3\ S)^I$ and $f \in (\leq 2\ S)^I$. Thus, d cannot have an RS-successor.

This proves that $A \sqsubseteq_{gfp,T} (\forall RS.\bot)$, and thus, $A \sqsubseteq_{gfp,T} (\forall RS.P)$. Consequently, $A \sqsubseteq_{gfp,T} B$. Because of $L_{\mathcal{A}_T}(B,P) \not\subseteq L_{\mathcal{A}_T}(A,P)$, the example shows that due to excluding words (RS in the example) the right-hand side of (1) is not necessary for subsumption w.r.t. $\mathcal{FLN}^r$.

Using E_A, we are able to characterize subsumption w.r.t. $\mathcal{FLN}^r$-terminologies and the gfp-semantics as follows:

Theorem 11 (subsumption).
Let T be an $\mathcal{FLN}^r$-terminology, $\mathcal{A}_T$ the corresponding semi-automaton, and A, B atomic concepts occurring in T. Then we have: $A \sqsubseteq_{gfp,T} B$ iff

1. $L(B,P) \subseteq L(A,P) \cup E_A$ for all primitive concepts P in T;
2. $L(B, (\geq l\ R)) \subseteq \bigcup_{r \geq l} L(A, (\geq r\ R)) \cup E_A$ for all $\geq$-restrictions of the form $(\geq l\ R)$ in T where $l > 0$; and
3. $L(B, (\leq l\ R)) \subseteq \bigcup_{r \leq l} L(A, (\leq r\ R)) \cup (E_A \cdot R^{-1})$ for all $\leq$-restrictions of the form $(\leq l\ R)$ in T. □

Proof sketch of Theorem 11 (for details see (Küsters, 1997)): Using that $W \in E_A$ implies $A \sqsubseteq_{gfp,T} \forall W.\bot$ and that $W \in (E_A \cdot R^{-1})$ implies $A \sqsubseteq_{gfp,T} \forall W.(\leq 0\ R)$ (thus, $A \sqsubseteq_{gfp,T} \forall W.(\leq l\ R)$ for all $l \geq 0$), the if direction of the claim is easy to show. The non-trivial part of the proof is the only-if direction. The idea is as follows: If one of the inclusions does not hold, then we define an extended canonical gfp-model I such that $d_0 \in A^I \setminus B^I$. For this purpose, the canonical model for A and d_0 is extended: If an inclusion is invalid because of the word W, i.e., this word is contained in the left-hand side but not in the right-hand side of the inclusion relation, the tree of the canonical model with root d_0 is extended such that there is a path in the tree starting at d_0 labeled with W. If W violates the inclusion relation of a $\leq$-restriction $(\leq l\ R)$, then the path with label W additionally has to be extended by $(l+1)$ R-successors; more precisely by $(l+1)$ minus the already existing R-successors. Finally, this tree must be completed such that the condition formulated in Theorem 3 is satisfied w.r.t. A and d_0. Note that, because of the extension of the canonical model, these conditions initially need not be satisfied. Formally, the extended model is defined as follows:

Definition 12 (extended canonical gfp-model).
Let T be an $\mathcal{FLN}^r$-terminology, $\mathcal{A}_T$ the corresponding semi-automaton, A an atomic concept in T, W a word in Σ^*, $r \in \mathbb{N}$ and $R \in \Sigma$. For $r = 0$ we denote the *extended primitive canonical interpretation* by $J' = J(A, d_0, W)$; for $r > 0$ it is denoted by $J' = J(A, d_0, W, R, r)$. Let $U_1, U_2, U_3, \ldots$ be an enumeration of all words in Σ^* where $U_i \neq U_j$ for $i \neq j$ and $|U_i| \leq |U_j|$ for all $i < j$. Let $J = J(A, d_0)$ be the primitive canonical interpretation (introduced in Definition 6) and $I = I(A, d_0)$ the corresponding gfp-model of T. We define J' inductively as follows:

J_0: If W is required by A, then there is an individual $d_1 \in dom(J)$ such that $d_1 \in W^I(d_0)$ (see above). If $(r - |R^J(d_1)|) \geq 0$, then let $k := r - |R^J(d_1)|$ else $k := 0$. Let $f_1, \ldots, f_k$ be new, pairwise distinct individuals. We define:

$dom(J'_0) := dom(J) \dot\cup \{f_1, \ldots, f_k\}$; $S^{J'_0} := S^J$ for all roles $S \neq R$; $R^{J'_0} := R^J \dot\cup \{(d_1, f_i) \mid 1 \leq i \leq k\}$.

If W is not required by A, then there is a prefix U of W of maximal length which is required by A. Consequently, there is a $d_1 \in dom(J)$ such that $d_0 U^J d_1$. Furthermore, since W is not required by A, there is a $V \in \Sigma^+$, $V = R_1 \cdots R_n$, $n \geq 1$, where $W = UV$. Let $k := r$, and $d_2, d_3, \ldots, d_{n+1}, f_1, \ldots, f_k$ be new, pairwise distinct individuals. We define:

$dom(J'_0) := dom(J) \dot\cup \{d_2, \ldots, d_{n+1}, f_1, \ldots, f_k\}$; $S^{J'_0} := S^J \dot\cup \{(d_i, d_{i+1}) \mid 1 \leq i \leq n,\ S = R_i\} \dot\cup \{(d_{n+1}, f_i) \mid 1 \leq i \leq k,\ S = R\}$ for all roles S in T.

The extensions of the primitive concepts are defined only for J'.

J'_{i+1}: We define $M \subseteq dom(J'_i) \times \Sigma \times (\mathbb{N} \setminus \{0\})$ by $(g, S, m) \in M$ iff $g \in U_{i+1}^{J'_i}(d_0)$, $U_{i+1} \in L(A, (\geq m\ S))$, $m \geq 1$, m maximal with this property (i.e., there is no $m' > m$ such that $U_{i+1} \in L(A, (\geq m'\ S))$), and $|S^{J'_i}(g)| < m$. Let $(g_1, S_1, m_1), \ldots, (g_s, S_s, m_s)$ denote the triples in M, $s \geq 0$. Intuitively, M contains a triple (g, S, m) if for g in the corresponding tree to J'_i

additionally $(m - |S^{J'_i}(g)|)$ S-successors must be added in order to satisfy the conditions for the $\geq$-restrictions in Theorem 3 w.r.t. d_0 and A.

For this purpose, let $l_j := |S_j^{J'_i}(g_j)|$ (note that $l_j < m_j$) for all $1 \leq j \leq s$ and $D := \{g_1^1, \ldots, g_{m_1-l_1}^1, \ldots, g_1^s, \ldots, g_{m_s-l_s}^s\}$ a set of new, pairwise distinct individuals. We define:

$dom(J'_{i+1}) := dom(J'_i) \dot{\cup} D$; $S^{J'_{i+1}} := S^{J'_i} \dot{\cup} \bigcup_{1 \leq j \leq s, S=S_j} \{(g_j, g_1^j), \ldots, (g_j, g_{m_j-l_j}^j)\}$ for all roles S in T.

Now the extended primitive canonical interpretation J' is defined as follows:

$dom(J') := \bigcup_{i \in I\!N} dom(J'_i)$; $S^{J'} := \bigcup_{i \in I\!N} S^{J'_i}$ for all roles S in T; for all primitive concepts P and individuals d let $d \in P^{J'}$ iff there is a $V' \in L(A, P)$ such that $d \in V'^{J'}(d_0)$.

The *extended canonical gfp-models* $I(A, d_0, W)$ and $I(A, d_0, W, R, r)$ are the gfp-models of T and $J' = J(A, d_0, W)$ and $J' = J(A, d_0, W, R, r)$, respectively. $\diamond$

Now, if one of the inclusions in Theorem 11, 1. or 2. is violated by W, i.e., W is contained in the left-hand side but not in the right-hand side, then it can be shown that $d_0 \in A^{I'} \setminus B^{I'}$ where $I' := I(A, d_0, W)$. Furthermore, if in 3. the inclusion for the $\leq$-restriction $(\leq l\ R)$ is violated by W, then it can be shown that $d_0 \in A^{I'} \setminus B^{I'}$ where $I' := I(A, d_0, W, R, l+1)$. Since the proof that I' is the appropriate counter model for the subsumption of A and B is lengthy and technical, it is omitted here (see (Küsters, 1997) for details).

The inclusions in Theorem 11 can be decided by a decision procedure for inclusions of regular languages known from automata theory. Due to lack of space, we restrict our attention to the inclusions for primitive concepts.

In automata theory, the inclusion $L_1 \subseteq L_2$ of two regular languages is usually reduced to the emptiness problem of the finite automaton corresponding to the language $L_1 \cap \overline{L_2}$. It is known that this problem can be decided using space polynomial in the size of the automata $\mathcal{A}_1$, $\mathcal{A}_2$ corresponding to L_1 and L_2, respectively. Such an algorithm must cope with the complement $\overline{L_2}$ of L_2, and therefore, simulates the powerset automaton of L_2, which may be exponential in the size of $\mathcal{A}_2$.

In order to apply this technique from automata theory, we have to show that the language E_A is regular:

Definition 13 (reaching an exclusion set).
Let T be an $\mathcal{FLN}^r$-terminology and $\mathcal{A}_T$ the corresponding semi-automaton without word-transitions. There is an *exclusion set reachable by a word* $W \in \Sigma^*$ starting from the atomic concept A if there is a prefix V of W such that $next_\varepsilon(A, V)$ is an exclusion set. $\diamond$

Now, E_A can be described as follows (Küsters, 1997):

Lemma 14.
Let T be an $\mathcal{FLN}^r$-terminology, $\mathcal{A}_T$ the corresponding semi-automaton without word-transitions, and A an atomic concept in T. Then the following holds:

$$E_A = \{W \in \Sigma^* \mid \text{an exclusion set is reachable by } W \text{ starting from } A\}.$$

$\square$

Using this lemma, one can construct a finite automaton $\mathcal{B}$ corresponding to E_A (Küsters, 1997). Roughly speaking, this automaton is an extension of the powerset automaton of $\mathcal{A}_T$ where the exclusion sets are certain states in $\mathcal{B}$. This shows that E_A is regular. Furthermore, $L(\mathcal{B}, P) \subseteq L(A, P) \cup E_A$ can be decided in space polynomial in the size of $\mathcal{A}_T$ since the powerset automaton of $\mathcal{A}_T$, which is used to cope with $\overline{L(A, P) \cup E_A}$, can be simulated (see (Küsters, 1997) for details and for the inclusions in 2. and 3. of Theorem 11).

PSPACE-completeness of subsumption in general $\mathcal{FL}_0$-terminologies (Baader, 1996b) implies

Proposition 15.
Subsumption in general $\mathcal{ALN}$-terminologies with respect to gfp-semantics is PSPACE-complete. $\square$

Descriptive semantics: For the descriptive semantics, the notion "exclusion" is specified as in Definition 10. However, Theorem 11 has to be extended by the following additional condition:

4. For all defined concepts C in T and all infinite paths of the form $B, U_0, C, U_1, C, U_2, C, \ldots$ there is a $k \geq 0$ such that $U_0 \cdots U_k \in L(A, C) \cup E_A$.

Apart from the union with E_A on the right-hand side of the element relationship, this condition coincides with the one for $\mathcal{FL}_0$ in (Baader, 1996b) (for $\mathcal{FL}_0$ the set E_A is empty).

Based on the power-set automata and the exclusion sets of $\mathcal{A}_T$ one can decide condition 4. using polynomial space (Küsters, 1997). For $\mathcal{FL}_0$-terminologies and the descriptive semantics PSPACE-completeness of subsumption is still an open problem. However, PSPACE-completeness of inconsistency for the descriptive semantics in $\mathcal{ALN}$-terminologies provides us with PSPACE-completeness of subsumption in $\mathcal{ALN}$-terminologies w.r.t. descriptive semantics since inconsistency can be reduced to subsumption.

Lfp-semantics: Similar to the notion "exclusion set" we have to extend the notion "exclusion" for lfp-semantics due to the special treatment necessary for infinite role chains. Beside the condition for exclusion given in Definition 10, for the lfp-semantics a word $W \in \Sigma^*$ also excludes the atomic concept A if

> there is a (finite or infinite) word $X \in U(A)$ as well as a word $V \in \Sigma^*$ which is prefix of W and X such that X is required by A starting from V.

In addition, the notion of "exclusion" must be extended to infinite words $W \in \Sigma^\omega$: W *excludes* A if there is a finite prefix of W which excludes A or if $W \in U(A)$. As for the other two semantics, we refer to the set of finite A-excluding words by E_A. Moreover, the set of finite and infinite A-excluding words is denoted $E_{A,\omega}$. Now, for the lfp-semantics Theorem 11 is extended by the following condition:

4. $U(B) \subseteq U(A) \cup E_{A,\omega}$.

Again, based on the power-set automata and the exclusion sets of $\mathcal{A}_T$ one can decide this condition using polynomial space (Küsters, 1997). Furthermore, PSPACE-completeness of subsumption for $\mathcal{FL}_0$-terminologies with respect to lfp-semantics implies PSPACE-completeness of subsumption even for $\mathcal{ALN}$-terminologies.

6 $\mathcal{SLN}$-SCHEMAS AND $\mathcal{ALN}$-TERMINOLOGIES

In (Buchheit et al., 1994), terminologies are divided into a schema and a view part. The schema merely restricts the number of admissible models of a terminology. Therefore, the meaning of schemas is captured by descriptive semantics. In the view part of a terminology, concepts are *defined* with the help of schema concepts. For this reason, fixed-point semantics is used for this part.

Knowledge and database engineers are interested in validity of schemas as well as subsumption w.r.t. schemas. In (Buchheit et al., 1994), $\mathcal{SL}_{dis}$-schemas have been introduced, which can express constraints frequently occurring in the static part of object-oriented database schemas. Furthermore, a special PSPACE-decision algorithm has been developed for deciding (local) validity of these schemas. In (Küsters, 1997), it is shown that inconsistency, validity, and subsumption for $\mathcal{SLN}$-schemas (and hence, for $\mathcal{SL}_{dis}$-schemas) can be reduced to corresponding problems for $\mathcal{ALN}$-terminologies. As mentioned above, this allows us to prove some hardness results for $\mathcal{ALN}$-terminologies.

In the following, we have a closer look at this reduction by showing how to construct a terminology T_S from a given schema S and by considering the relationship between T_S and S w.r.t. inconsistency and subsumption. First, however, we must introduce schemas formally.

An *$\mathcal{SLN}$-schema* S consists of a finite set of concept inclusions, which define necessary conditions for concepts, and role inclusions, which define (simple) necessary conditions for roles. To be more precise, let A, B be atomic concepts, R a role name, and n a non-negative integer. Concept inclusions are of the form $A \sqsubseteq C$ where C denotes a concept of the form $B \mid \neg B \mid \forall R.B \mid (\geq n\ R) \mid (\leq n\ R)$, and role inclusions are of the form $R \sqsubseteq A \times B$. An atomic concept A with at least one concept inclusion of the form $A \sqsubseteq C$ in S is called *defined*, and otherwise *primitive*. Analogously, we refer to *defined* and *primitive* roles. Note that here we allow for primitive negation only, i.e., for concept inclusions $A \sqsubseteq \neg B$ where B is primitive. Using so-called *$\mathcal{ALN}$-schemas*, which allow for arbitrary $\mathcal{ALN}$-concepts (rather than "flat" $\mathcal{ALN}$-concepts) on the right-hand side of a concept inclusion, does not increase the expressive power (Küsters, 1997). Furthermore, $\mathcal{SL}_{dis}$-schemas introduced in (Buchheit et al., 1994) coincide with $\mathcal{SLN}$-schemas (w.r.t. expressive power) apart from the fact that they only allow for number restrictions of the form $(\leq 1\ R)$ and $(\geq 1\ R)$. Finally, $\mathcal{AL}$-schemas correspond to $\mathcal{SL}_{dis}$-schemas in which role inclusions are not allowed.

An interpretation I is a *model* of S if all concept inclusions $A \sqsubseteq C$ and all role inclusions $R \sqsubseteq A \times B$ in S are satisfied, i.e., $A^I \subseteq C^I$ and $R^I \subseteq A^I \times B^I$. A concept A is *consistent* w.r.t. S if there is a model I of S with $A^I \neq \emptyset$. A schema S is *locally valid* if every atomic concept in S is consistent. We call a schema S *valid* if there is a model I with $A^I \neq \emptyset$ for *every* atomic concept A in S. The following fact (Buchheit et al., 1994; Küsters, 1997) can be used to reduce validity of schemas to consistency of terminologies:

> An $\mathcal{SLN}$-schema ($\mathcal{SL}_{dis}$-schema) is valid iff it is locally valid. (2)

Now, we derive an $\mathcal{ALN}$-terminology T_S from an $\mathcal{SLN}$-schema S such that T_S behaves like S w.r.t. consistency and subsumption. Before defining T_S, we transform S into a schema S' that does not contain role inclusions. Role inclusions only have to be taken into account if role successors are required, otherwise they can be dispensed with. The formal definition of S' is as follows:

The schema S' contains all concept inclusions of S of the form $A \sqsubseteq B$, $A \sqsubseteq \neg B$, $A \sqsubseteq \forall R.B$, and $A \sqsubseteq (\leq n\, R)$. For every $A \sqsubseteq (\geq n\, R) \in S$, $n \geq 1$, we distinguish two cases: (a) R is not defined in S; (b) R is defined in S. In case (a), $A \sqsubseteq (\geq n\, R)$ is contained in S'. In case (b), S' contains $A \sqsubseteq (\geq n\, R) \sqcap \forall R.C_2 \sqcap C_1$ for every $R \sqsubseteq C_1 \times C_2 \in S$. The schema S' contains no other inclusions than these, in particular, no role inclusions.

Definition 16 (the terminology T_S).
Let S be an $\mathcal{SLN}$-schema and S' as defined above. For every defined concept A in S a concept definition for A in T_S is constructed as follows:
Let $A \sqsubseteq C_1, \ldots, A \sqsubseteq C_n$ be all concept inclusions for the defined concept A in S'. Let $\overline{A}$ be a new (primitive) concept. Then the concept definition for A in T_S is of the form $A = \overline{A} \sqcap C_1 \sqcap \cdots \sqcap C_n$. ◇

Obviously, one can compute T_S in time linear in the size of S. The relationship between S and T_S is described by the following theorem (Küsters, 1997):

Theorem 17.
Let S be an $\mathcal{SLN}$-schema and T_S the corresponding $\mathcal{ALN}$-terminology. For all atomic concepts A and B in S we have:

- A is S-consistent iff A is T_S-consistent w.r.t. descriptive semantics (iff A is T_S-consistent w.r.t. gfp-semantics).
- $A \sqsubseteq_S B$ iff $A \sqsubseteq_{T_S} B$ (iff $A \sqsubseteq_{gfp,T_S} B$). □

Consequently using the complexity results presented in section 4 and 5, consistency—and with (2) also (local) validity—as well as subsumption w.r.t. descriptive and gfp-semantics of $\mathcal{SLN}$- and $\mathcal{ALN}$-schemas are decidable using polynomial space. This result not only extends results for complexity upper bounds of $\mathcal{SL}_{dis}$-schemas to $\mathcal{SLN}$-schemas, but also uses well-known results and techniques from automata theory, rather than defining special graphs as in (Buchheit et al., 1994). Furthermore, it clarifies the relationship between schemas and terminologies for the considered languages: $\mathcal{ALN}$-terminologies are at least as expressive as $\mathcal{SLN}$-schemas w.r.t. the important inference problems. Finally, it should be noted that the complexity of subsumption w.r.t. schemas is only due to testing consistency of concepts (Buchheit et al., 1994). Hence, disallowing concept forming operators that enable the definition of inconsistent concepts (number restrictions and primitive negation) makes reasoning tractable, whereas for $\mathcal{FL}_0$-terminologies subsumption is already PSPACE-complete (w.r.t. fixed-point semantics).

7 CONCLUSION

Cyclic terminologies are a natural way to describe the terminological knowledge of a problem domain. Unlike the case of first-order predicate logic, the transitive closure of relations is expressible using the gfp-semantics. Furthermore, as already pointed out, cyclic definitions occur frequently, e.g., in constraints for semantic and object-oriented data models. The previous section has shown that general $\mathcal{ALN}$-terminologies are expressive enough to capture important parts of such constraints.

Neither the descriptive nor the fixed-point semantics capture the intuition of a cyclic terminology in any case (see, e.g., (Nebel, 1990a; Baader, 1996b; Küsters, 1997)). Therefore, it is crucial to gain a more profound understanding of the intuition of these semantics, in order to make decision easier which semantics is to prefer in the representation task at hand. For this reason, automata theoretic characterizations have been provided for the three semantics with respect to general $\mathcal{FLN}$-terminologies. The characterizations of the three semantics themselves are easy extensions of the characterizations for $\mathcal{FL}_0$-terminologies (Baader, 1996b).

Using this automata theoretic descriptions, we have also proved characterizations of the important inference problems inconsistency and subsumption for $\mathcal{FLN}$ where non-trivial extensions in comparison with $\mathcal{FL}_0$-terminologies are necessary. Moreover, these descriptions have been used to derive decision algorithms and complexity results for inconsistency and subsumption.

Finally, we have shown that the important inference problems for $\mathcal{SLN}$- and $\mathcal{SL}_{dis}$-schemas can be reduced to corresponding problems for terminologies. This reduction provides us with automata theoretic decision algorithms for these problems, and it clarifies the expressive power of the considered schemas and terminologies; in addition, it extends the results for $\mathcal{SL}_{dis}$-schemas (Buchheit et al., 1994) to $\mathcal{SLN}$-schemas, which allow for arbitrary number-restrictions.

Future work will extend the characterizations presented here in order to derive decision algorithms for consistency and instance w.r.t. knowledge bases consisting of cyclic terminologies and *A-Boxes*. Furthermore, it has turned out that the automata-theoretic characterizations can be used to handle inference problems like computing the least common subsumer of concepts (LCS) and the most specific concept of individuals (MSC). Cyclic terminologies are necessary in this context since w.r.t. A-Boxes with role cycles the MSC of an individual may not be describable in terms of acyclic terminologies.

Acknowledgment

I am very grateful to Franz Baader for his helpful comments and suggestions as well as Ralf Molitor for valuable discussions.

References

Aït-Kaci, H. (1984). *A Lattice-Theoretic Approach to Computations Based on a Calculus of Partially Ordered Type Structures*. PhD thesis, University of Pennsylvania, PA.

Baader, F. (1990). Terminological cycles in KL-ONE-based knowledge representation languages. In *Proceedings of AAAI-90*, pages 621–626.

Baader, F. (1996a). A formal definition for the expressive power of terminological knowledge representation languages. *J. of Logic and Computation*, 6(1):33–54.

Baader, F. (1996b). Using automata theory for characterizing the semantics of terminological cycles. *Annals of Mathematics and Artificial Intelligence*, 18(2–4):175–219.

Bergamaschi, S. and Sartori, C. (1992). On taxonomic reasoning in conceptual design. *ACM Trans. on Database Systems*, 17(3):385–442.

Borgida, A. (1995). Description logics in data management. *IEEE Trans. on Knowledge and Data Engineering*, 7(5):671–682.

Buchheit, M., Donini, F., Nutt, W., and Schaerf, A. (1994). Refining the structure of terminological systems: Terminology = Schema + View. In *Proceedings of the 12th National Conference of the American Association for Artificial Intelligence, AAAI-94*, Seattle.

Buchheit, M., Donini, F., Nutt, W., and Schaerf, A. (1997). A refined architecture for terminological systems: Terminology = Schema + View. *Artificial Intelligence*. To appear.

Calvanese, D. (1996). Reasoning with inclusion axioms in description logics: Algorithms and Complexity. In Wahlster, W., editor, *Proceedings of the 12th European Conference on Artificial Intelligence, ECAI-96*. John Wiley & Sons.

Calvanese, D., Lenzerini, M., and Nardi, D. (1994). A unified framework for class based representation formalisms. In *Proceedings of the Fourth International Conference on the Principles of Knowledge Representation and Reasoning (KR-94)*, pages 109–120. Morgan Kaufmann, Los Altos.

Giacomo, G. D. and Lenzerini, M. (1994). Concept languages with number restrictions and fixpoints, and its relationship with mu-calculus. In *Proceedings of the Eleventh European Conference on Artificial Intelligence, ECAI-94*, pages 411–415. John Wiley and Sons.

Giunchiglia, F. and Sebastiani, R. (1996). A sat-based decision procedure for $\mathcal{ALC}$. In *5th International Conference on Principles of Knowledge Representation and Reasoning - KR'96*. Morgan Kaufmann, Los Altos.

Hollunder, B. and Nutt, W. (1990). Subsumption algorithms for concept languages. Technical Report RR-90-04, German Research Center for Artificial Intelligence, Germany.

Küsters, R. (1997). Characterizing the semantics of terminological cycles in $\mathcal{ALN}$ using finite automata. Technical Report LTCS-97-04, RWTH Aachen, Germany. ftp://www-lti.informatik.rwth-aachen.de/pub/papers/1997/Kuesters-LTCS-97-04.ps.gz.

MacGregor, R. (1992). What's needed to make a description logic a good KR citizen. In *Working Notes of the AAAI Fall Symposium on Issues on Description Logics: Users meet Developers*, pages 53–55.

Nebel, B. (1987). On terminological cycles. In *KIT Report 58*, Technische Universität Berlin.

Nebel, B. (1990a). Reasoning and revision in hybrid representation systems. In *Lecture Notes in Computer Science 422*. Springer Verlag, Berlin.

Nebel, B. (1990b). Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235–249.

Nebel, B. (1991). Terminological cycles: Semantics and computational properties. In Sowa, J., editor, *Formal Aspects of Semantic Networks*, pages 331–361. Morgan Kaufmann, San Mateo.

Schild, K. (1994). Terminological cycles and the propositional μ-calculus. In *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning, KR'94*, pages 509–520, Bonn, Germany. Morgan Kaufmann.

Description Logics and Graph-based Languages

Satisfiability problem in description logics with modal operators

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

Michael Zakharyaschev
Institute of Applied Mathematics
Russian Academy of Sciences
Miusskaya Square 4
125047 Moscow, Russia
e-mail: mz@spp.keldysh.ru

Abstract

The paper considers the standard concept description language $\mathcal{ALC}$ augmented with various kinds of modal operators which can be applied to concepts and axioms. The main aim is to develop methods of proving decidability of the satisfiability problem for this language and apply them to description logics with most important temporal and epistemic operators, thereby obtaining satisfiability checking algorithms for these logics. We deal with the possible world semantics under the constant domain assumption and show that the expanding and varying domain assumptions are reducible to it. Models with both finite and arbitrary constant domains are investigated. We begin by considering description logics with only one modal operator and then prove a general transfer theorem which makes it possible to lift the obtained results to many systems of polymodal description logic.

1 INTRODUCTION

Description (or terminological) logics have been developed and used[1] as a formalism for representing knowledge about static application domains. Having stemmed from real working systems like KL-ONE and its successors, they proved to be a successful compromise between expressibility and effectiveness.

In a description logic system, the knowledge of an application domain is represented in the form of conceptual and assertional axioms. The former introduce the relevant terminology—complex concepts defined in terms of atomic ones and binary relations (roles) between objects with the help of certain constructors. And the latter describe facts about some concrete objects in the domain in terms of concept and role instances. Although the existing description languages provide a wide choice of constructors (see e.g. Baader et al., 1990, Donini et al., 1996), usually they are intended to represent only static knowledge and are not able to express various dynamic aspects such as time-dependence, beliefs of different agents, obligations, etc., which are regarded to be important ingredients in modeling intelligent agents.

For example, in every standard description language we can define a concept "good car" as, say, a car with an airconditioner:

$$\textit{good car} = \textit{car} \wedge \exists \textit{part.airconditioner}. \qquad (1)$$

However, we have no means to represent the subtler knowledge that only John believes (1) to be the case, while Mary does not think so:

$$[\textit{John believes}](1) \wedge \neg[\textit{Mary believes}](1).$$

Nor can we express the fact that (1) holds now but in future the notion of a good car may change (since, for instance, all cars will have airconditioners):

$$(1) \wedge \langle \textit{eventually} \rangle \neg(1).$$

A way to bridge this gap seems to be quite clear. One can simply combine a description language with a suitable modal language treating belief, temporal, deontic or some other intensional operators. But one has to be careful, for such a combination may ruin the balance between expressibility and effectiveness, as it happened with too powerful pure description languages (see e.g. Schmidt-Schauß, 1989 or Patel-Schneider, 1989).

[1]See e.g. (Brachman and Schmolze 1985), (Borgida et al. 1989), (Baader and Hollunder 1991), and (Donini et al. 1996) for more references.

There is a number of parameters that determine the design of a modal extension of a given description language.

(I) First, modal operators can be applied to different kinds of well-formed expressions of the description language.

One can apply them only to conceptual and assertional axioms thereby forming new axioms of the form:

$$[\textit{John believes}](\textit{good car} = \textit{car} \wedge \exists \textit{part.airconditioner}),$$

$$[\textit{Mary believes}]\,\langle \textit{eventually}\rangle\,(\textit{John is rich}).$$

Modal operators can be applied to concepts in order to form new ones:

$$[\textit{John believes}]\ \textit{expensive}$$

(i.e., the concept of all objects John believes to be expensive) or

$$\textit{human being} \wedge \exists \textit{child}.[\textit{Mary believes}]\ \langle \textit{eventually}\rangle\ \textit{good student}$$

(i.e., the concept of all human beings with a child which Mary believes to be eventually a good student).

By allowing applications of modal operators to both concepts and axioms we obtain expressions of the form

$$[\textit{John believes}](\textit{good car} = [\textit{Mary believes}]\ \textit{good car})$$

(i.e., John believes that a car is good if and only if Mary thinks so).

Finally, one can supplement the options above with modal operators applicable to roles. For example, using the temporal operator [*always*] (in future) and the role *loves*, we can form the new role [*always*] *loves* (which is understood as a relation between objects x and y that holds if and only if x will always love y) to say

$$\textit{John} : \exists[\textit{always}]\ \textit{loves.woman}$$

(i.e., John will always love the very same woman (but perhaps not only her), which is not the same as $\textit{John} : [\textit{always}]\exists \textit{loves.woman}$).

(II) All these languages are interpreted with the help of the possible world semantics in which the accessibility relations between worlds treat the modal operators,[2] and the worlds themselves consist of domains in which the concepts, role names and object names of the description component are interpreted.

The properties of the modal operators are determined by the conditions we impose on the corresponding accessibility relations. For example, by imposing no condition at all we obtain what is known as the minimal normal modal logic **K**—although of definite theoretical interest, it does not have the properties required to model operators like [*agent A knows*], ⟨*eventually*⟩, etc. Transitivity of the accessibility relation for agent A's knowledge means what is called the positive introspection (A knows what he knows), Euclideannes corresponds to the negative introspection (A knows what he does not know), reflexivity reflects that only true facts are known to A (for more information and further references consult e.g. Halpern and Moses, 1992). In the temporal case, depending on the application domain we may assume time to be linear and discrete (i.e., the usual strict ordering of the natural numbers), or branching, or dense, etc. (see van Benthem, 1996).

(III) Another important parameter is the number of modal operators we need in our language and, respectively, the number of the corresponding accessibility relations. If we deal with multi-agent epistemic logic then every agent A gives rise to the operator [*agent A believes*]. If we also want to capture the development of beliefs in time, we should add the corresponding temporal operator. Note that certain combinations of "harmless" modalities may result in a logic of extremely high complexity (see e.g. Spaan, 1993).

(IV) When connecting worlds—that is ordinary models of the pure description language—by accessibility relations, we are facing the problem of connecting their objects. Depending on the particular application, we may assume worlds to have arbitrary domains (the ***varying domain assumption***), or we may assume that the domain of a world accessible from a world w contains the domain of w (the ***expanding domain assumption***), or that all the worlds share the same domain (the ***constant domain assumption***). Consider, for instance, the following axioms:

$$\neg[\textit{agent A knows}](\textit{unicorn} = \bot),$$

$$([\textit{agent A knows}]\neg \textit{unicorn}) = \top.$$

The former means that agent A does not know that unicorns do not exist, while according to the latter, for every existing object, A knows that it is not a unicorn. Such a situation can be modeled under the expanding domain assumption, but these two formulas cannot be simultaneously satisfied in a model with constant domains.

[2]E.g. [*agent A believes*] φ is regarded to be true in a world w iff φ is true in all the worlds agent A considers to be possible in w or, in other words, accessible from w via the relation interpreting agent A's beliefs.

(V) Following (Calvanese 1996), one can distinguish between models with finite and infinite domains. In many applications of pure description logics finite domains are preferable: after all the real world a knowledge base is talking about is finite. (For instance, when the domain consists of employees of a company then certainly we should assume it to be finite.) However, if we are dealing with time and temporal operators, it is natural to assume that with time passing potentially infinitely many different objects may appear in the application domain of the knowledge base. Note that the finite domain assumption does not mean that models are finite.

(VI) Finally, one should take into account the difference between *rigid* and *non-rigid designators*. In our context, the former are the object names interpreted by the same objects in every world in the model under consideration, while the latter are those whose interpretation is not fixed. Again the choice between these depends on the application domain: if the knowledge base is talking about employees of a company then the name *John Smith* should probably denote the same person no matter what world we consider, while *President of the company* may refer to different persons in different worlds. For a more detailed discussion consult e.g. (Fitting 1993) or (Kripke 1980).

The following kinds of description modal logics have been studied in the literature. Laux (1994) constructed a multi-agent logic of belief in which the belief operators apply only to axioms, the accessibility relations are transitive, serial and Euclidean, domains are constant and of arbitrary size, and designators are rigid. Schild (1993) introduced description logics with temporal operators applicable only to concepts and interpreted in models with linear and branching discrete unbounded time under the constant domain assumption and rigid designators. Baader and Laux (1995) consider a language in which modal operators can be applied to both axioms and concepts; they are interpreted in models with arbitrary accessibility relations under the expanding domain assumption. Baader and Ohlbach (1995) use modal operators as role constructors, but exclude object names and assertions from the language.

The languages of Schild (1993) and Laux (1994) present no serious technical difficulties: the satisfiability problem for both of them is reducible to the satisfiability problem in the well-known propositional modal logic (in the former case this was observed by Schild himself and the latter is treated by Theorem 7 below). On the other hand, the unrestricted use of modal operators to form new roles may lead to undecidable logics even under very natural conditions for the other parameters, as was proved by Baader and Ohlbach (1995).

The language of Baader and Laux (1995) appears to be sufficiently expressive and yet manageable. However, it was analyzed only in the abstract case of **K**-type modalities. More interesting for applications are modal operators with explicit temporal or epistemic interpretations to which the decision procedure of Baader and Laux is not extended. Besides, their technique works only under the expanding domain assumption. In general, the case of constant domains turns out to be much harder. First, there are description logics lacking the finite model property under the constant domain assumption but enjoying it if expanding domains are allowed (see Remark 10 below). And second, one can actually reduce the case of expanding or varying domains to that of constant domains (see Theorem 6).

Baader and Laux (1995) did not consider specially models with finite domains. Actually, in their case there is no need to distinguish between the variants of finite and infinite domains: as will be shown below, the sets of formulas satisfiable in models with arbitrary accessibility relations are the same no matter which of the two variants is adopted. However, these sets become different if we consider linear temporal models or models whose accessibility relations are reflexive and transitive (see Theorem 9). A similar situation arises in pure description logic when one extends the expressive power in such a way that the resultant logic does not have the finite model property (see e.g. De Giacomo and Lenzerini, 1994). In this case the set of formulas satisfiable in finite domains does not coincide with the set of formulas satisfiable in arbitrary domains.

The aim of this paper is to develop methods of proving decidability of the satisfiability problem for the description language with modal operators and apply them to most important systems. We will consider modal description logics with the following parameters.

1. The modal operators can be applied to concepts and axioms, but not to roles.
2. The language is interpreted in models with the accessibility relations satisfying most conditions of the standard nomenclature for the belief and temporal operators (in modal logic they correspond to the systems K, S5, KD45, S4, S4.3, GL, GI.3 and the tense logic of discrete linear unbounded time).
3. We begin by considering description logics with

only one modal operator and then prove a general transfer theorem which makes it possible to lift the obtained results to many systems of polymodal description logic.

4. We adopt the constant domain assumption and show that the varying domain assumption as well as the expanding domain assumption are reducible to it.

5. Both finite and arbitrary constant domains are considered.

6. Designators are assumed to be rigid.

(The standard way of proving decidability in modal logic by using a variant of the filtration technique does not work for the logics under consideration. First, the filtration of worlds often conflicts with the constant domain assumption (which is not the case when expanding domains are allowed). And second, not all our logics enjoy the finite model property.)

Although our underlying description language is the standard $\mathcal{ALC}$, the obtained results can be extended to languages with more expressive power, for instance, to $\mathcal{ALC}$ enriched with number restrictions or transitive reflexive closure. The proof of this claim as well as various other proofs are omitted and can be found in the full paper.

2 SYNTAX AND SEMANTICS

Definition 1 (alphabet) The *primitive symbols* of the modal concept description language $\mathcal{ALC}_{\mathcal{M}}$ are:

- concept names: $C_0, C_1, \dots$;
- role names: $R_0, R_1, \dots$;
- object names: $a_0, a_1, \dots$;
- the booleans (say, $\wedge$, $\neg$, $\top$), modal operators $\Diamond_0, \Diamond_1, \dots$, and the relativized existential quantifier $\exists R_i$, for every role name R_i.

Other standard logical connectives are defined in the usual way. For instance, $C \to D$ is an abbreviation for $\neg(C \wedge \neg D)$, $\bot$ for $\neg\top$, and $\Box_i$ for $\neg\Diamond_i\neg$.

Definition 2 (concept) Concepts are defined inductively as follows: all concept names as well as $\top$ are (*atomic*) *concepts*, and if C, D are concepts, R is a role name, and $\Diamond_i$ a modal operator in our language then $C \wedge D$, $\neg C$, $\Diamond_i C$, $\exists R.C$ are *concepts.*

Definition 3 (formula) Let C and D be concepts, R a role name and a, b object names. Then expressions of the form $C = D$, aRb, $a : C$ are (*atomic*) *formulas.* If φ and ψ are formulas then so are $\Diamond_i\varphi$, $\neg\varphi$, and $\varphi\wedge\psi$.

Note that in the definition above we did not impose any restriction on the form of conceptual and assertional axioms. (Baader and Laux (1995) consider, for instance, only atomic formulas prefixed by sequences of modal operators.) This will have no affect on our decidability results as far as we do not touch on the complexity of the decision algorithms.

By $md(\varphi)$, the *modal depth* of a formula φ, we mean the length of the longest chain of nested modal operators in φ (including those in the concepts occurring in φ); $\Box^{\leq m}\varphi$ is the conjunction of all distinct formulas which are obtained by prefixing to φ a sequence of $\leq m$ operators $\Box_0, \Box_1, \dots$ (in arbitrary order). For instance,

$$\Box^{\leq 2}\varphi = \varphi \wedge \Box_0\varphi \wedge \Box_1\varphi \wedge \dots \wedge \Box_0\Box_1\varphi \wedge \Box_1\Box_0\varphi \wedge \dots$$

Denote by $con\varphi$, $rol\varphi$ and $ob\varphi$ the sets of all concepts, role names and object names occurring in φ, respectively; $sub\varphi$ is the set of all subformulas in φ.

We remind the reader that models of a pure modal language are based on Kripke frames, structures of the form $\mathfrak{F} = \langle W, \lhd_0, \lhd_1, \dots \rangle$ in which each $\lhd_i$ is a binary (accessibility) relation on the set of worlds W. What is going on inside the worlds is of no importance. Models of $\mathcal{ALC}_{\mathcal{M}}$ are also constructed on Kripke frames; however, in this case their worlds are models of $\mathcal{ALC}$.

Definition 4 (model) A *model* of $\mathcal{ALC}_{\mathcal{M}}$ based on a frame $\mathfrak{F} = \langle W, \lhd_0, \lhd_1, \dots \rangle$ is a pair $\mathfrak{M} = \langle \mathfrak{F}, I \rangle$ in which I is a function associating with each $w \in W$ a structure

$$I(w) = \left\langle \Delta^{I(w)}, R_0^{I(w)}, \dots, C_0^{I(w)}, \dots, a_0^{I(w)}, \dots \right\rangle,$$

where $\Delta^{I(w)}$ is a non-empty set of objects, the *domain* of w, $R_i^{I(w)}$ are binary relations on $\Delta^{I(w)}$, $C_i^{I(w)}$ subsets of $\Delta^{I(w)}$, and $a_i^{I(w)}$ are objects in $\Delta^{I(w)}$ such that $a_i^{I(w)} = a_i^{I(v)}$, for any $v, w \in W$.

One can distinguish between three types of models: those with *constant*, *expanding*, and *varying domains.* In models with constant domains $\Delta^{I(v)} = \Delta^{I(w)}$, for all $v, w \in W$. In models with expanding domains $\Delta^{I(v)} \subseteq \Delta^{I(w)}$ whenever $v \lhd_i w$, for some i. And models with varying domains are just arbitrary models.

Definition 5 (satisfaction) For a model $\mathfrak{M} = \langle \mathfrak{F}, I \rangle$ and a world w in it, the *value* $C^{I(w)}$ of a concept C in w and the *truth-relation* $(\mathfrak{M}, w) \models \varphi$ (or simply $w \models \varphi$, if $\mathfrak{M}$ is understood) are defined inductively in the following way:

1. $\top^{I(w)} = \Delta^{I(w)}$ and $C^{I(w)} = C_i^{I(w)}$, for $C = C_i$;
2. $(C \wedge D)^{I(w)} = C^{I(w)} \cap D^{I(w)}$;
3. $(\neg C)^{I(w)} = \Delta^{I(w)} - C^{I(w)}$;
4. $x \in (\Diamond_i C)^{I(w)}$ iff $\exists v \rhd_i w \; x \in C^{I(v)}$;
5. $x \in (\exists R_i.C)^{I(w)}$ iff $\exists y \in C^{I(w)} \; x R_i^{I(w)} y$;
6. $w \models C = D$ iff $C^{I(w)} = D^{I(w)}$;
7. $w \models a : C$ iff $a^{I(w)} \in C^{I(w)}$;
8. $w \models aR_ib$ iff $a^{I(w)} R_i^{I(w)} b^{I(w)}$;
9. $w \models \Diamond_i \varphi$ iff $\exists v \rhd_i w \; v \models \varphi$;
10. $w \models \varphi \wedge \psi$ iff $w \models \varphi$ and $w \models \psi$;
11. $w \models \neg\varphi$ iff $w \not\models \varphi$.

A formula φ is *satisfiable* in a class of models $\mathcal{M}$ if there is a model $\mathfrak{M} \in \mathcal{M}$ and a world w in $\mathfrak{M}$ such that $w \models \varphi$.

In this paper our main concern is to find out whether there exist algorithms for checking satisfiability of formulas in several important classes of models. Other standard inference problems (concept satisfiability, subsumption, instance checking, consistency) are reducible to the satisfiability problem. The entailment problem can also be reduced to it, at least for the classes of models considered below: this is clear for the local consequence—$\Gamma \models_{\mathcal{M}} \varphi$ iff $(\mathfrak{M}, w) \models \Gamma \Rightarrow (\mathfrak{M}, w) \models \varphi$, for every $\mathfrak{M} \in \mathcal{M}$ and every world w in $\mathfrak{M}$—in this case $\Gamma \models_{\mathcal{M}} \varphi$ iff $\neg(\bigwedge \Gamma \rightarrow \varphi)$ is not satisfiable in $\mathcal{M}$. For the global consequence—$\Gamma \models^*_{\mathcal{M}} \varphi$ iff $\mathfrak{M} \models \Gamma \Rightarrow \mathfrak{M} \models \varphi$, for every $\mathfrak{M} \in \mathcal{M}$—we have $\Gamma \models^*_{\mathcal{M}} \varphi$ iff $\neg(\bigwedge \Gamma \wedge \Box \bigwedge \Gamma \rightarrow \varphi)$ is not satisfiable in $\mathcal{M}$ when models in $\mathcal{M}$ are transitive, and the class of all models is treated similarly to Theorem 3.57 of (Chagrov and Zakharyaschev 1997).

With every class $\mathcal{C}$ of Kripke frames (the number of accessibility relations in which corresponds to the number of modal operators in $\mathcal{ALC}_{\mathcal{M}}$) we associate the classes $\mathcal{M}(\mathcal{C})$, $\mathcal{M}^e(\mathcal{C})$, and $\mathcal{M}^v(\mathcal{C})$ of all models of $\mathcal{ALC}_{\mathcal{M}}$ based on frames in $\mathcal{C}$ and having constant, expanding and varying domains, respectively; $\mathcal{M}_{fin}(\mathcal{C})$ will denote the class of models based on frames in $\mathcal{C}$ and having constant finite domains. The set of formulas satisfiable in a class of models $\mathcal{M}$ will be denoted by $Sat\mathcal{M}$.

We will use special names for certain classes of frames with one accessibility relation. Namely,

- $\mathcal{K}$ will stand for the class of all frames (with arbitrary accessibility relations),
- $\mathcal{GL}$ for the class of transitive frames without infinite ascending chains (in other words, transitive Noetherian frames),
- $\mathcal{GL}.3$ for the class of transitive Noetherian frames which are linear (i.e., $u \lhd v \vee v \lhd u \vee u = v$),
- $\mathcal{S}5$ will stand for the class of frames with the universal relations, i.e., $u \lhd v$ for all u and v (this class is often regarded to be a good model for explicit knowledge),
- $\mathcal{S}4$ for the class of frames with transitive reflexive relations (i.e., quasi-ordered frames),
- $\mathcal{S}4.3$ for the class of linear quasi-ordered frames,
- $\mathcal{KD}45$ will stand for the class of transitive, serial ($\forall u \exists v \; u \lhd v$) and Euclidean ($u \lhd v \wedge u \lhd w \rightarrow v \lhd w$) frames (this class is often regarded to be a good model for explicit beliefs that are not necessarily true), and
- $\mathcal{N}$ for the frame $\langle \mathbb{N}, < \rangle$, where $\mathbb{N}$ is the set of natural numbers.

By $\mathcal{K}_n$ ($\mathcal{GL}_n$, etc.) we denote the classes of frames with n arbitrary (respectively, n transitive Noetherian, etc.) accessibility relations.

Our strategy is to consider first the unimodal case ($n = 1$) and then lift the obtained results to the polymodal one by proving a general transfer theorem for independent joins of logics.

Let us start, however, with two simple observations. First, it turns out that the satisfiability problem for models with expanding and varying domains can be reduced to the satisfiability problem for models with constant domains. To show this, we introduce a concept ex the intended meaning of which is to contain in each world precisely those objects that are assumed to exist (under the expanding or varying domain assumption) in this world. By relativizing all concepts and formulas to the concept ex, one can simulate varying and expanding domains using constant ones.

Theorem 6 *If $Sat\mathcal{M}(\mathcal{C})$ is decidable, $\mathcal{C}$ a class of frames, then $Sat\mathcal{M}^e(\mathcal{C})$ and $Sat\mathcal{M}^v(\mathcal{C})$ are also decidable.*

Proof Given a formula φ, let ex be a concept name which does not occur in φ. By induction on the construction of a concept C we define its relativization

$C{\downarrow}\mathsf{ex}$:

$$\begin{aligned} C_i{\downarrow}\mathsf{ex} &= C_i \wedge \mathsf{ex},\ C_i \text{ a concept name,} \\ (C \wedge D){\downarrow}\mathsf{ex} &= (C{\downarrow}\mathsf{ex}) \wedge (D{\downarrow}\mathsf{ex}), \\ (\neg C){\downarrow}\mathsf{ex} &= \mathsf{ex} \wedge \neg(C{\downarrow}\mathsf{ex}), \\ (\exists R.C){\downarrow}\mathsf{ex} &= \mathsf{ex} \wedge \exists R.(C{\downarrow}\mathsf{ex}), \\ (\Diamond_i C){\downarrow}\mathsf{ex} &= \mathsf{ex} \wedge \Diamond_i(C{\downarrow}\mathsf{ex}). \end{aligned}$$

The relativization of φ is defined inductively as follows:

$$\begin{aligned} (aRb){\downarrow}\mathsf{ex} &= aRb \wedge (a:\mathsf{ex}) \wedge (b:\mathsf{ex}), \\ (a:C){\downarrow}\mathsf{ex} &= a:(C{\downarrow}\mathsf{ex}), \\ (C=D){\downarrow}\mathsf{ex} &= ((C{\downarrow}\mathsf{ex}) = (D{\downarrow}\mathsf{ex})), \\ (\neg\varphi){\downarrow}\mathsf{ex} &= \neg(\varphi{\downarrow}\mathsf{ex}), \\ (\varphi \wedge \psi){\downarrow}\mathsf{ex} &= (\varphi{\downarrow}\mathsf{ex}) \wedge (\psi{\downarrow}\mathsf{ex}), \\ (\Diamond_i\varphi){\downarrow}\mathsf{ex} &= \Diamond_i(\varphi{\downarrow}\mathsf{ex}). \end{aligned}$$

Suppose now that $\mathfrak{F} = \langle W, \lhd_0, \dots\rangle$ is a frame and $m = md(\varphi)$. Then φ is satisfiable in a model based on $\mathfrak{F}$ and having varying domains iff the formula

$$\varphi' = \varphi{\downarrow}\mathsf{ex} \wedge \Box^{\leq m}(\neg(\mathsf{ex} = \bot) \wedge \bigwedge_{a \in ob\varphi} a:\mathsf{ex})$$

is satisfiable in a model based on $\mathfrak{F}$ and having constant domains. Indeed, assuming that φ is satisfied in a model $\mathfrak{M} = \langle \mathfrak{F}, I\rangle$ with varying domains and that

$$I(w) = \left\langle \Delta^{I(w)}, R_0^{I(w)}, \dots, C_0^{I(w)}, \dots, a_0^{I(w)}, \dots \right\rangle,$$

for $w \in W$, we construct a model $\mathfrak{N} = \langle \mathfrak{F}, J\rangle$ with constant domains by defining $J(w)$ as

$$\langle \bigcup_{w \in W} \Delta^{I(w)}, R_0^{I(w)}, \dots, C_0^{I(w)}, \dots, \mathsf{ex}^{J(w)}, a_0^{I(w)}, \dots\rangle,$$

where $\mathsf{ex}^{J(w)} = \Delta^{I(w)}$. It is readily checked by induction that for any $\psi \in sub\varphi$ and any $w \in W$, $(\mathfrak{M}, w) \models \psi$ iff $(\mathfrak{N}, w) \models \psi{\downarrow}\mathsf{ex}$. Thus φ' is satisfied in $\mathfrak{N}$.

Conversely, suppose φ' is satisfied in a world v in a model $\mathfrak{N} = \langle \mathfrak{F}, J\rangle$ with constant domains and that

$$J(w) = \left\langle \Delta, R_0^{J(w)}, \dots, C_0^{J(w)}, \dots, \mathsf{ex}^{J(w)}, a_0^{J(w)}, \dots \right\rangle,$$

for $w \in W$. Consider the model $\mathfrak{M} = \langle \mathfrak{F}, I\rangle$ in which

$$I(w) = \left\langle \mathsf{ex}^{J(w)}, R_0^{I(w)}, \dots, C_0^{I(w)}, \dots, a_0^{J(w)}, \dots \right\rangle,$$

where $R_i^{I(w)}$ and $C_i^{I(w)}$ are the restrictions of $R_i^{J(w)}$ and $C_i^{J(w)}$ to $\mathsf{ex}^{J(w)}$, respectively, for every w accessible from v by $\leq m$ steps[3], and $I(w) = J(w)$ for all the other worlds w in $\mathfrak{F}$. Since $(\mathfrak{N}, v) \models \Box^{\leq m}\neg(\mathsf{ex} = \bot)$, the domains of worlds in $\mathfrak{M}$ are not empty. One can show by induction that for every $\psi \in sub\varphi$, $(\mathfrak{N}, v) \models \psi{\downarrow}\mathsf{ex}$ iff $(\mathfrak{M}, v) \models \psi$ (here we use the fact, well-known in modal logic, that the truth-value of φ in v depends only on the worlds accessible by $\leq m$ steps from v).

[3]I.e., $v \, \Delta_i \, v_1 \, \Delta_j \cdots \Delta_k \, v_{n-1} \, \Delta_l \, v_n$ for some $n \leq m$ and some $i, j, \dots, k, l$.

The case of expanding domains is considered analogously by adding to φ' under $\Box^{\leq m}$ one more conjunct $(\mathsf{ex} \to \Box^{\leq 1}\mathsf{ex}) = \top$. ❑

Theorem 6 gives us grounds for restricting attention only to models with constant domains. So in the remaining part of the paper we adopt the constant domain assumption.

Our second observation concerns the satisfiability problem for the set $\mathcal{ALC}^-_{\mathcal{M}}$ of formulas in $\mathcal{ALC}_{\mathcal{M}}$ which contain no concepts of the form $\Diamond_i C$. By extending the technique of Finger and Gabbay (1992) to modal description logics in which modal operators apply only to formulas one can prove the following:

Theorem 7 *If the modal logic characterized by a class of frames $\mathcal{C}$ is decidable, then the sets of formulas in $\mathcal{ALC}^-_{\mathcal{M}}$ that are satisfiable in the classes $\mathcal{M}(\mathcal{C})$ and $\mathcal{M}_{fin}(\mathcal{C})$ coincide and the satisfiability problem for them is decidable.*

Note also that this restricted language does not feel any difference between constant and non-constant domains.

From now on till Section 6 we will be considering the concept description language $\mathcal{ALC}_{\mathcal{M}}$ with only one modal operator $\Diamond$ (and its dual $\Box$).

3 LOGICS WITHOUT THE FINITE MODEL PROPERTY

In pure modal logic, the classes of frames introduced in Section 2 determine decidable logics, which is usually established by proving their finite model property (see e.g. Chagrov and Zakharyaschev, 1997). However, this way of proving decidability does not go through for all corresponding modal description logics.

Definition 8 (FMP) Say that the set $Sat\mathcal{M}(\mathcal{C})$, $\mathcal{C}$ a class of frames, has the *finite model property* (FMP, for short) if every formula in $Sat\mathcal{M}(\mathcal{C})$ is satisfiable in a finite model in $\mathcal{M}(\mathcal{C})$. $Sat\mathcal{M}(\mathcal{C})$ has the *bounded FMP* (BMP, for short) if there is an effective function $f : \mathbb{N} \mapsto \mathbb{N}$ such that every formula φ in $Sat\mathcal{M}(\mathcal{C})$ is satisfiable in a model from $\mathcal{M}(\mathcal{C})$ with at most $f(|\varphi|)$

worlds and objects, $|\varphi|$ the length (say, the number of symbols) of φ.

It should be clear that if $Sat\mathcal{M}(\mathcal{C})$ has BMP and the set of finite frames in $\mathcal{C}$ is recursive then there ia an algorithm deciding whether a given formula is satisfiable in $\mathcal{M}(\mathcal{C})$. If $Sat\mathcal{M}(\mathcal{C})$ has FMP then clearly $Sat\mathcal{M}(\mathcal{C}) = Sat\mathcal{M}_{fin}(\mathcal{C})$. But the converse does not hold in general as follows from the existence of pure modal logics without FMP.

Theorem 9 *For any class $\mathcal{C} \in \{\mathcal{S}4, \mathcal{S}4.3, \mathcal{N}, \mathcal{GL}.3\}$, $Sat\mathcal{M}(\mathcal{C}) \supsetneq Sat\mathcal{M}_{fin}(\mathcal{C})$.*

Proof For a formula ψ, let $\Box^+\psi = \psi \wedge \Box\psi$ and let φ_1 be the conjunction of the following formulas:

$$a : C, \quad \Box^+((C \to \Box C) = \top),$$

$$\Box^+(\exists R.\neg C = \top), \quad \Box^+((\neg C \to \Diamond C) = \top).$$

One can readily check that φ_1 is satisfied in the models $\langle\langle \mathbb{N}, <\rangle, I\rangle$ and $\langle\langle \mathbb{N}, \leq\rangle, I\rangle$ in which, for every $n \in \mathbb{N}$,

$$I(n) = \langle \mathbb{N}, R^n, C^n, a^n\rangle,$$

where $R^n = \mathbb{N} \times \mathbb{N}$, $C^n = \{0, \dots, n\}$, $a^n = 0$. It is not hard to see, however, that φ_1 cannot be satisfied in any model based on $\langle \mathbb{N}, <\rangle$ or on a frame in $\mathcal{S}4$ and having a finite domain. It follows that $Sat\mathcal{M}(\mathcal{C}) \supsetneq Sat\mathcal{M}_{fin}(\mathcal{C})$, for $\mathcal{C} \in \{\mathcal{S}4, \mathcal{S}4.3, \mathcal{N}\}$.

Now take φ_2 to be the conjunction of the following three formulas:

$$\Diamond(C \wedge \neg\Diamond C) = \top, \quad \Box((C \to \exists R.\neg C) = \top),$$

$$\Box((C \to \neg\exists R.\Diamond C) = \top).$$

It is readily checked that φ_2 is true at the root of the model $\langle\langle W, \lhd\rangle, I\rangle$ in which $W = \{0, 1, \dots, \omega\}$, $i \lhd j$ iff $i > j$, for $i, j \in W$ (so the frame $\langle W, \lhd\rangle$ is transitive, linear and Noetherian), and for every $n \in W$,

$$I(n) = \left\langle \mathbb{N}, R^{I(n)}, C^{I(n)} \right\rangle,$$

where $C^{I(n)} = \{n\}$, for $n < \omega$, $C^{I(\omega)} = \emptyset$, and $0R^{I(n)}1R^{I(n)}2R^{I(n)}\dots$. However, φ_2 is not satisfiable in any transitive linear Noetherian model with a finite domain. ❑

Remark 10 It is of interest to note that (i) φ_2 is satisfied in a non-linear Noetherian model with only three worlds and two objects (see Theorem 24), (ii) $Sat\mathcal{M}^e(\mathcal{GL}.3)$ has FMP (only two worlds are enough to satisfy φ_2 under the expanding domain assumption), and (iii) $Sat\mathcal{M}_{fin}(\mathcal{GL}.3)$ has BMP (see Theorem 25).

4 DECIDABILITY WITHOUT BMP

Our aim in this section is to present an algorithm for checking satisfiability in models based on $\langle \mathbb{N}, <\rangle$. Let us fix an arbitrary formula φ.

Definition 11 (quasiworld) A *quasiworld* for φ is a structure of the form

$$\mathfrak{w} = \langle X, R_0^{\mathfrak{w}}, \dots, C_0^{\mathfrak{w}}, \dots, (\Diamond D_0)^{\mathfrak{w}}, \dots, a_0^{\mathfrak{w}}, \dots\rangle,$$

where X is a finite set, $R_i^{\mathfrak{w}} \subseteq X \times X$ for every $R_i \in rol\varphi$, $C_i^{\mathfrak{w}} \subseteq X$ for every concept name C_i in φ, $(\Diamond D_i)^{\mathfrak{w}} \subseteq X$ for every $\Diamond D_i \in con\varphi$, and $a_i^{\mathfrak{w}} \in X$ for every $a_i \in ob\varphi$. The *value* $C^{\mathfrak{w}}$ of a concept $C \in con\varphi$ in $\mathfrak{w}$ is computed as in Definition 5, but with item 4 replaced by the following: $C^{\mathfrak{w}} = (\Diamond D_i)^{\mathfrak{w}}$, for $C = \Diamond D_i$.

Now consider a structure

$$\mathfrak{m} = \langle \mathfrak{w}_1, \dots, \mathfrak{w}_k | \mathfrak{w}_{k+1}, \dots, \mathfrak{w}_l\rangle, \tag{2}$$

in which $\mathfrak{w}_i$, $1 \leq i \leq l$, are quasiworlds for φ with domains X_i. Define a function $h : \mathbb{N} \mapsto \{1, \dots, l\}$ by taking $h(i) = i$ for $1 \leq i \leq l$ and $h(l + m) = k + 1 + \mathrm{mod}_{l-k}(m - 1)$, for $m > 0$; that is h returns the sequence $1, \dots, k, k+1, \dots, l, k+1, \dots, l, \dots$.

Definition 12 (run) A *run* in $\mathfrak{m}$ is any sequence $r = x_1, x_2, \dots$ such that $x_i \in X_{h(i)}$ and, for every concept $\Diamond C \in con\varphi$ and every $i < \omega$,

(a) $x_i \in (\Diamond C)^{\mathfrak{w}_{h(i)}}$ iff $x_j \in C^{\mathfrak{w}_{h(j)}}$ for some $j > i$.

The ith element of a run r will be denoted by $r(i)$, the quasiworld $\mathfrak{w}_{h(i)}$ by $\mathfrak{w}(i)$ and its domain by $X(i)$ (thus $r(i) \in X(i)$). Any two elements $x = r(i)$ and $y = r(i + 1)$ of a run r satisfy the following condition:

$$\forall \Diamond C \in con\varphi \; (x \in (\Diamond C)^{\mathfrak{w}(i)} \Leftrightarrow$$
$$y \in C^{\mathfrak{w}(i+1)} \cup (\Diamond C)^{\mathfrak{w}(i+1)}).$$

A pair $x \in X(i)$, $y \in X(i + 1)$ satisfying it will be called *suitable*.

Definition 13 (quasimodel) A structure $\mathfrak{m}$ of the form (2) is a *quasimodel* for φ if the following conditions hold:

(b) for every $a \in ob\varphi$, $r_a = a^{\mathfrak{w}(1)}, a^{\mathfrak{w}(2)}, \dots$ is a run in $\mathfrak{m}$;

(c) for every $i < \omega$ and every $x \in X(i)$, there is a run r in $\mathfrak{m}$ such that $r(i) = x$.

Example 14 The structure $\mathfrak{m} = \langle\; |\mathfrak{w}\rangle$ in which

$$\mathfrak{w} = \langle X, R^{\mathfrak{w}}, C^{\mathfrak{w}}, (\Diamond C)^{\mathfrak{w}}, (\Diamond\neg C)^{\mathfrak{w}}, a^{\mathfrak{w}}\rangle,$$

where $X = \{x, y, z\}$, $R^{\mathfrak{w}} = X \times X$, $C^{\mathfrak{w}} = \{x\}$, $(\Diamond C)^{\mathfrak{w}} = X$, $(\Diamond \neg C)^{\mathfrak{w}} = \{z\}$, and $a^{\mathfrak{w}} = x$ is a quasimodel for the formula φ_1 constructed in the proof of Theorem 9. The sequences

$$r_1 = x, x, x, x, x, \ldots, \quad r_2 = y, x, x, x, x, \ldots,$$

$$r_3 = z, y, x, x, x, \ldots, \quad r_4 = z, z, y, x, x, \ldots$$

are runs in $\mathfrak{m}$, while $r = z, z, z, z, z, \ldots$ is not a run because $z \in (\Diamond C)^{\mathfrak{m}(1)}$ but $z \notin C^{\mathfrak{m}(i)}$ for any $i < \omega$.

It is worth noting that given a structure of the form (2), we can effectively decide whether it is a quasimodel for φ. For we have the following:

Lemma 15 *A structure $\langle \mathfrak{w}_1, \ldots, \mathfrak{w}_k | \mathfrak{w}_{k+1}, \ldots, \mathfrak{w}_l \rangle$, in which all $\mathfrak{w}_i$ are quasiworlds for φ, is a quasimodel for φ iff*

(i) *for every $i \leq l$ and every $y \in X(i+1)$ there exists $x \in X(i)$ such that the pair x, y is suitable; in particular, for any $a \in ob\varphi$, every pair of adjacent elements in the sequence $a^{\mathfrak{w}(1)}, \ldots, a^{\mathfrak{w}(l+1)}$ is suitable;*

(ii) *for every $i \leq l$ and every $x_0 \in X(i)$ there is*

$$n \leq k + |con\varphi| \cdot (l - k) \cdot |X_{k+1}| \cdot \ldots \cdot |X_l|$$

and there are objects $x_j \in X(i + j)$, for $j = 1, \ldots, n$, such that

$$\forall \Diamond C \in con\varphi \; (x_0 \in (\Diamond C)^{\mathfrak{w}(i)} \Rightarrow \\ \exists m \in \{1, \ldots, n\} \; x_{i+m} \in C^{\mathfrak{w}(i+m)}), \quad (3)$$

with every pair x_j, x_{j+1}, for $0 \leq j < n$, being suitable; in particular, for every $a \in ob\varphi$ and every $i \leq l$,

$$\forall \Diamond C \in con\varphi \; (a^{\mathfrak{w}(i)} \in (\Diamond C)^{\mathfrak{w}(i)} \Rightarrow \\ \exists m \leq l - k \; a^{\mathfrak{w}(i+m)} \in C^{\mathfrak{w}(i+m)}).$$

Proof ($\Leftarrow$) To construct a run through $x_m \in X(m)$, $m < \omega$, we first take objects $x_i \in X(i)$, for $i < m$, such that every pair of adjacent elements in the sequence $x_1, \ldots, x_m$ is suitable—this can be done by (i). Then using (ii) we select a sequence $x_m, \ldots, x_{m+n}$ such that every pair of adjacent elements in it is suitable and $x_m \in (\Diamond C)^{\mathfrak{w}(m)}$ only if $x_{m+i} \in C^{\mathfrak{w}(m+i)}$ for some $i \leq n$. After that we select by (ii) such a sequence starting from $x_{m+n} \in X(m + n)$, and so on. It is readily seen that the resulting sequence $x_1, \ldots, x_m, \ldots, x_{m+n}, \ldots$ is a run in $\mathfrak{m}$.

($\Rightarrow$) That (i) holds follows immediately from (b), (c) and the definition of a run. To prove (ii), notice first that since some run in $\mathfrak{m}$ comes through $x_0 \in X(i)$, there is a sequence $x_j \in X(i+j)$, $j = 1, \ldots, n$, satisfying (3) and containing only suitable pairs x_j, x_{j+1}. So the problem is to bound n by the constant mentioned in the formulation of the lemma. And this can be done by deleting certain redundant segments from the sequence $x_0, \ldots, x_n$ using the obvious fact that to reach x_j from x_i, $k+1 \leq i < j \leq n$, (via suitable pairs of objects) one needs not more than $(l-k) \cdot |X_{k+1}| \cdot \ldots \cdot |X_l|$ elements. ❑

The truth-relation $\mathfrak{w}(i) \models \psi$ in a quasimodel $\mathfrak{m}$ is computed in the same way as in Definition 5, but with item 9 replaced by the following: $\mathfrak{w}(i) \models \Diamond\psi$ iff $\mathfrak{w}(j) \models \psi$ for some $j > i$.

Given a quasimodel $\mathfrak{m}$ for φ of the form (2), we can construct a standard model $\mathfrak{M} = \langle \langle \mathbb{N}, < \rangle, I \rangle$ in the following way. Its domain Δ consists of all runs in $\mathfrak{m}$ and, for every $n \in \mathbb{N}$,

$$I(n) = \left\langle \Delta, R_0^{I(n)}, \ldots, C_0^{I(n)}, \ldots, r_{a_0}, \ldots \right\rangle,$$

where $rR_i^{I(n)}r'$ iff $r(n)R_i^{\mathfrak{w}(n)}r'(n)$, and $r \in C_i^{I(n)}$ iff $r(n) \in C_i^{\mathfrak{w}(n)}$. By a straightforward induction one can show that for all $C \in con\varphi$, $\psi \in sub\varphi$, $n \in \mathbb{N}$ and $r \in \Delta$, we have $r \in C^{I(n)}$ iff $r(n) \in C^{\mathfrak{w}(n)}$, and $n \models \psi$ iff $\mathfrak{w}(n) \models \psi$ (condition (a) ensures that $r \in (\Diamond D)^{I(n)}$ iff $r(n) \in (\Diamond D)^{\mathfrak{w}(n)}$ and condition (c) guarantees that $r \in (\exists R_i.D)^{I(n)}$ iff $r(n) \in (\exists R_i.D)^{\mathfrak{w}(n)}$). Thus, a formula φ is satisfiable in $\mathcal{M}(\mathcal{N})$ whenever φ is satisfiable in some quasimodel for φ.

To prove the converse, for a model $\mathfrak{M} = \langle \langle \mathbb{N}, < \rangle, I \rangle$ satisfying φ and having a domain Δ, we construct a quasimodel representing $\mathfrak{M}$ modulo φ.

Definition 16 (types) The *type* of an object x in a world w of $\mathfrak{M}$ (relative to φ) is the set

$$t_w^{\mathfrak{M}}(x) = \{C \in con\varphi : x \in C^{I(w)}\}.$$

The *type* of w in $\mathfrak{M}$ (relative to φ) is the triple

$$T^{\mathfrak{M}}(w) = \langle \{t_w^{\mathfrak{M}}(x) : x \in \Delta\}, \\ \{\psi \in sub\varphi : w \models \varphi\}, \{\langle a, t \rangle : t_w^{\mathfrak{M}}(a^{I(w)}), a \in ob\varphi \rangle \rangle.$$

We will omit the superscript $\mathfrak{M}$ and write simply $t_w(x)$ and $T(w)$ if understood.

Every model contains at most $2^{|con\varphi|}$ objects of pairwise distinct types in every world and at most

$$\sharp(\varphi) = 2^{2^{|con\varphi|}} \cdot 2^{|sub\varphi|} \cdot |ob\varphi| \cdot 2^{|con\varphi|}$$

worlds having pairwise distinct types.

With every world i in $\mathfrak{M}$ we associate the quasiworld

$$\mathfrak{w}_i = \langle X_i, R_0^{\mathfrak{w}_i}, \dots, C_0^{\mathfrak{w}_i}, \dots, (\Diamond D_0)^{\mathfrak{w}_i}, \dots, a_0, \dots \rangle,$$

where X_i contains the objects $a \in ob\varphi$ from Δ[4] and also one representative $z \notin ob\varphi$ from each class $[x]_i = \{y \in \Delta : t_i(x) = t_i(y)\}$, if such z exists (so $|X_i| \leq \flat(\varphi) = 2^{|con\varphi|} + |ob\varphi|$), $xR_j^{\mathfrak{w}_i}y$ iff either one of x, y is not in $ob\varphi$ and $x'R_j^{I(i)}y'$ for some $x' \in [x]_i$, $y' \in [y]_i$, or $x, y \in ob\varphi$ and $xR_j^{I(i)}y$, $x \in C_j^{\mathfrak{w}_i}$ iff $x \in C_j^{I(i)}$, and $x \in (\Diamond D_j)^{\mathfrak{w}_i}$ iff $x \in (\Diamond D_j)^{I(i)}$.

Consider the structure $\mathfrak{m} = \langle \mathfrak{w}_1, \dots, \mathfrak{w}_k | \mathfrak{w}_{k+1}, \dots, \mathfrak{w}_l \rangle$ in which each $\mathfrak{w}_i$ is the quasiworld associated with the world i in $\mathfrak{M}$, $1 \leq i \leq l$, k is the minimal number such that $T(k+1)$ occurs infinitely often in the sequence $T(k+1), T(k+2), \dots$, and l is the minimal number such that $T(k+1) = T(l+1)$ and the following conditions (d) and (e) hold:

(d) $\forall \Diamond C \in con\varphi\ \forall a \in ob\varphi\ (a \in (\Diamond C)^{\mathfrak{w}_{k+1}} \Leftrightarrow \exists i \in \{k+2, \dots, l\}\ a \in C^{\mathfrak{w}_i})$,

(e) for every $x_{k+1} \in X_{k+1}$ there are $x_{k+i} \in X_{k+i}$, $i = 2, \dots, l-k$, such that every pair x_{k+j}, x_{k+j+1} is suitable and

$$\forall \Diamond C \in con\varphi\ (x_{k+1} \in (\Diamond C)^{\mathfrak{w}_{k+1}} \Leftrightarrow \exists i \in \{k+2, \dots, l\}\ x_i \in C^{\mathfrak{w}_i}).$$

By Lemma 15, $\mathfrak{m}$ is a quasimodel for φ. Indeed, (i) follows from the fact that every pair $x \in [y]_i$, $x' \in [y]_{i+1}$, for $y \in \Delta$, is suitable. And to show (ii) it suffices to take $n = 2l - k - i$ and the sequence

$$x_i \in X_i, \dots, x_l \in X_l, x_{l+1} \in X_{k+1}, \dots, x_{2l} \in X_l$$

such that every pair of adjacent elements in $x_i, \dots, x_l, x_{l+1}$ is suitable and $x_{l+1}, \dots, x_{2l}$ satisfies (e). It is easily checked by induction that $\mathfrak{m}$ satisfies φ.

We show now that by deleting some quasiworlds from $\mathfrak{m}$ one can construct a quasimodel satisfying φ and containing not more than some effectively computed number of quasiworlds.

In the "linear" part $\mathfrak{w}_1, \dots, \mathfrak{w}_k$ of $\mathfrak{m}$ we delete all the quasiworlds $\mathfrak{w}_{i+1}, \dots, \mathfrak{w}_j$ such that $T(i) = T(j)$, for $i < j \leq k$. By Lemma 15, the resulting structure is again a quasimodel satisfying φ. Thus we may assume that $T(i) \neq T(j)$ whenever $1 \leq i \neq j \leq k$, and so $k \leq \sharp(\varphi)$.

Let us consider now the "cyclic" part $\mathfrak{w}_{k+1}, \dots, \mathfrak{w}_l$. For every $x \in X_{k+1}$, fix a sequence $s_x = x_{k+2}, \dots, x_l$ satisfying (e) (for $x = a \in ob\varphi$ we take $s_x = a^{\mathfrak{w}_{k+2}}, \dots, a^{\mathfrak{w}_l}$) and put $s_x(i) = x_i$, $i \in \{k+2, \dots, l\}$. There are at most $\flat(\varphi)$ sequences s_x satisfying (e). For each of them, say s_x, we mark (at most $|con\varphi|$) numbers $m, \dots, m'$ in the set $\{k+2, \dots, l\}$ such that $s_x(n) \in C^{\mathfrak{w}_n}$, for some $n \in \{m, \dots, m'\}$, whenever $x \in (\Diamond C)^{\mathfrak{w}_{k+1}}$. Let $m_1 < \dots < m_n$ be all marked numbers for all $x \in X_{k+1}$. We will keep the quasiworlds $\mathfrak{w}_{k+1}, \mathfrak{w}_{m_1}, \dots, \mathfrak{w}_{m_n}$ in our quasimodel. (Note that $n \leq |con\varphi| \cdot \flat(\varphi)$.) And if for $i \in \{k+2, \dots, l\} - \{m_1, \dots, m_n\}$ there is $j > i$ such that $T(i) = T(j)$ and

$$\{i, i+1, \dots, j-1\} \cap \{k+1, m_1, \dots, m_n\} = \emptyset$$

then we delete all the quasiworlds $\mathfrak{w}_i, \dots, \mathfrak{w}_{j-1}$ from $\mathfrak{m}$. The number of the remaining quasiworlds in the "cyclic" part does not exceed $|con\varphi| \cdot \flat(\varphi) \cdot \sharp(\varphi)$. Using Lemma 15 one can readily see that the resulting structure is a quasimodel satisfying φ.

Thus a formula φ is satisfiable in $\mathcal{M}(\mathcal{N})$ iff it is satisfiable in a quasimodel for φ of some effectively computable size. And the latter condition is effectively checked with the help of Lemma 15. Using similar (though technically more sophisticated) methods one can construct satisfiability checking algorithms for $\mathcal{M}(\mathcal{S}4)$, $\mathcal{M}(\mathcal{S}4.3)$ and $\mathcal{M}(\mathcal{GL}.3)$. Thus we obtain

Theorem 17 *The satisfiability problem for $\mathcal{M}(\mathcal{N})$, $\mathcal{M}(\mathcal{S}4)$, $\mathcal{M}(\mathcal{S}4.3)$ and $\mathcal{M}(\mathcal{GL}.3)$ is decidable.*

5 PROVING BMP

As we shall see in this section, all the sets $Sat\mathcal{M}(\mathcal{C})$, for $\mathcal{C} \in \{\mathcal{K}, \mathcal{KD}45, \mathcal{S}5, \mathcal{GL}\}$, have BMP. In principle, one can prove this by filtrating worlds through some suitable sets of formulas and duplicating certain objects in the filtrated worlds to comply with the constant domain assumption. It turns out, however, that actually the same result can be achieved by using the method of quasimodels we started developing above.

By quasimodels in this section we will mean certain frames of the form

$$\mathfrak{m} = \langle Q, \lhd \rangle \qquad (4)$$

in which Q is a set of quasiworlds for some formula φ and $\lhd$ a binary relation on Q. To give a precise definition we again require a notion of a run in $\mathfrak{m}$.

Definition 18 (run) A *run* in $\mathfrak{m} = \langle Q, \lhd \rangle$ is a set r which contains precisely one object from the domain $X_{\mathfrak{w}}$ of each quasiworld $\mathfrak{w} \in Q$—let us denote this object by $r(\mathfrak{w})$—and, for every $r(\mathfrak{u})$ and every $\Diamond C \in con\varphi$, we have $r(\mathfrak{u}) \in (\Diamond C)^{\mathfrak{u}}$ iff there is $r(\mathfrak{v}) \in C^{\mathfrak{v}}$ for some $\mathfrak{v} \rhd \mathfrak{u}$.

[4]Without loss of generality we may assume $a^{I(n)} = a$.

Definition 19 (quasimodel) A *quasimodel* for a formula φ is a frame $\mathfrak{m}$ of the form (4) such that

(f) for every $a \in ob\varphi$, $r_a = \{a^{\mathfrak{w}} : \mathfrak{w} \in Q\}$ is a run in $\mathfrak{m}$;

(g) every object in every quasiworld in $\mathfrak{m}$ belongs to some run in $\mathfrak{m}$.

The truth-relation $(\mathfrak{m}, \mathfrak{w}) \models \psi$ is defined similarly to Definition 5. Given a quasimodel $\mathfrak{m} = \langle Q, \lhd \rangle$ for φ, construct a standard model $\mathfrak{M} = \langle \mathfrak{m}, I \rangle$ by taking for each $\mathfrak{w} \in Q$

$$I(\mathfrak{w}) = \left\langle \Delta, R_0^{I(\mathfrak{w})}, \dots, C_0^{I(\mathfrak{w})}, \dots, a_0^{I(\mathfrak{w})}, \dots \right\rangle,$$

where Δ is the set of all runs in $\mathfrak{m}$, $rR_i^{I(\mathfrak{w})}r'$ iff $r(\mathfrak{w})R_i^{\mathfrak{w}}r'(\mathfrak{w})$, $r \in C_i^{I(\mathfrak{w})}$ iff $r(\mathfrak{w}) \in C_i^{\mathfrak{w}}$, and $a_i^{I(\mathfrak{w})} = r_{a_i}(\mathfrak{w})$. It is readily checked by induction that for all $C \in con\varphi$, $\psi \in sub\varphi$, $\mathfrak{w} \in Q$ and $r \in \Delta$, we have $r \in C^{I(\mathfrak{w})}$ iff $r(\mathfrak{w}) \in C^{\mathfrak{w}}$, and $(\mathfrak{M}, \mathfrak{w}) \models \psi$ iff $(\mathfrak{m}, \mathfrak{w}) \models \psi$.

$Sat\mathcal{M}(\mathcal{K})$: It is well known from modal logic (see e.g. Chagrov and Zakharyaschev, 1997) that every satisfiable purely modal formula φ can be satisfied in a finite intransitive tree of depth $\leq md(\varphi)$ and branching $\leq |sub\varphi|$. We remind the reader that a frame $\mathfrak{F} = \langle W, \lhd \rangle$ is called a *tree* if (i) $\mathfrak{F}$ is *rooted*, i.e., there is $w_0 \in W$ (a *root* of $\mathfrak{F}$) such that $w_0 \lhd^* w$ for every $w \in W$, where $\lhd^*$ is the transitive and reflexive closure of $\lhd$, and (ii) for every $w \in W$, the set $\{v \in W : v \lhd^* w\}$ is finite and linearly ordered by $\lhd^*$. The *depth* of a tree is the length of its longest branch. A tree $\mathfrak{F} = \langle W, \lhd \rangle$ is *intransitive* if every world v in $\mathfrak{F}$, save its root, has precisely one predecessor, i.e., $|\{u \in W : u \lhd v\}| = 1$, and the root w_0 is *irreflexive*, i.e., $\neg w_0 \lhd w_0$ (in fact, all worlds in an intransitive frame are irreflexive). Using the standard technique of modal logic one can prove the following

Lemma 20 *Every $\varphi \in Sat\mathcal{M}(\mathcal{K})$ is satisfiable in a model based on an intransitive tree of depth $\leq md(\varphi)$.*

Thus, to establish BMP of $Sat\mathcal{M}(\mathcal{K})$ it remains to show that trees of finite branching are enough to satisfy all formulas in $Sat\mathcal{M}(\mathcal{K})$ and to estimate the degree of branching.

Suppose a formula φ is satisfied in a model $\mathfrak{M} = \langle \mathfrak{F}, I \rangle$ based on an intransitive tree $\mathfrak{F} = \langle W, \lhd \rangle$ of depth $\leq md(\varphi)$ (but possibly with infinitely many branches). As in Section 4, with every world $w \in W$ we associate the quasiworld

$$\mathfrak{w} = \langle X_{\mathfrak{w}}, R_0^{\mathfrak{w}}, \dots, C_0^{\mathfrak{w}}, \dots, (\Diamond D_0)^{\mathfrak{w}}, \dots, a_0^{\mathfrak{w}}, \dots \rangle$$

for φ. (The associated quasiworlds will be denoted by the Gothic letters corresponding to the Roman letters denoting the worlds in $\mathfrak{F}$.) Let $\mathfrak{m} = \langle Q, \prec \rangle$ be the quasimodel for φ in which $Q = \{\mathfrak{w} : w \in W\}$ and $\mathfrak{u} \prec \mathfrak{v}$ iff $u \lhd v$. We are going to select (by induction) a subtree $\mathfrak{m}' = \langle Q', \prec' \rangle$ of $\mathfrak{m}$ which is also a quasimodel for φ and whose degree of branching is $\leq |con\varphi| \cdot \mathfrak{b}(\varphi) + |sub\varphi|$. The root of $\mathfrak{m}'$ is the root of $\mathfrak{m}$. Assume now that we have already selected a quasiworld $\mathfrak{v}$ for Q' and are looking for its successors. Consider an arbitrary object $x \in X_{\mathfrak{v}}$ and all the concepts $\Diamond D_i \in con\varphi$, for $i = 1, \dots, n$, such that $x \in (\Diamond D_i)^{\mathfrak{v}}$. By condition (g), there is a run r in $\mathfrak{m}$ containing x and such that $r(\mathfrak{u}_i) \in D_i^{\mathfrak{u}_i}$ for some $\mathfrak{u}_i \succ \mathfrak{v}$; if $x = a$, $a \in ob\varphi$, we use (f) instead of (g). Then we add $\mathfrak{u}_i$, for $i = 1, \dots, n$, to the quasimodel under construction as successors of $\mathfrak{v}$ and in the same manner consider the other objects in $X_{\mathfrak{v}}$. Also, for every $\Diamond\psi \in sub\varphi$ such that $\mathfrak{v} \models \Diamond\psi$, we add to our quasimodel one successor of $\mathfrak{v}$ in which ψ is true. Clearly, the total number of the added successors does not exceed $|con\varphi| \cdot \mathfrak{b}(\varphi) + |sub\varphi|$. To conclude the construction, we denote by Q' the set of all selected quasiworlds and define $\prec'$ to be the restriction of $\prec$ to Q. It is matter of routine to check that $\mathfrak{m}'$ is a quasimodel satisfying φ and

$$|Q'| \leq \sum_{n=0}^{md(\varphi)} (|con\varphi| \cdot \mathfrak{b}(\varphi) + |sub\varphi|)^n.$$

As a result we obtain the following

Theorem 21 *$Sat\mathcal{M}(\mathcal{K})$ has BMP and is decidable.*

$Sat\mathcal{M}(\mathcal{S}5)$: Let $\mathfrak{M} = \langle \mathfrak{F}, I \rangle$ be a model based on a frame $\mathfrak{F} = \langle W, W \times W \rangle$ and satisfying φ. In each class $[w] = \{v : T(w) = T(v)\}$ we select $\mathfrak{b}(\varphi)$ distinct representatives (if the number of worlds in $[w]$ is less than $\mathfrak{b}(\varphi)$ then we select all of them) and consider the structure $\mathfrak{m} = \langle Q, Q \times Q \rangle$ in which Q consists of the quasiworlds associated with those representatives (so $|Q| \leq \sharp(\varphi) \cdot \mathfrak{b}(\varphi)$). It is easily seen that $\mathfrak{m}$ is a quasimodel satisfying φ. Thus we have

Theorem 22 *$Sat\mathcal{M}(\mathcal{S}5)$ has BMP and is decidable.*

$Sat\mathcal{M}(\mathcal{KD}45)$: A frame in $\mathcal{KD}45$ is a non-degenerate cluster (i.e., a frame in $\mathcal{S}5$) possibly having one irreflexive predecessor (which in this case is the root of the frame). So, given a model $\mathfrak{M}$ based on such a frame and satisfying φ, we build a quasimodel $\mathfrak{m}$ for φ by taking the quasiworld associated with the root of $\mathfrak{M}$, if any (then it will be the irreflexive root of $\mathfrak{m}$), and the quasimodel for the cluster of $\mathfrak{M}$ constructed in precisely the same way as in the case of $Sat\mathcal{M}(\mathcal{S}5)$. This yields us

Theorem 23 *$Sat\mathcal{M}(\mathcal{KD}45)$ has BMP and is decidable.*

This technique can be also adopted to prove

Theorem 24 *$Sat\mathcal{M}(\mathcal{GL})$ has BMP and is decidable.*

We close this section with a decidability result under the finite constant domain assumption.

Theorem 25 *Let $\mathcal{C}$ be any of the following classes of frames: (i) all transitive frames, (ii) all transitive reflexive frames, (iii) all transitive linear frames, (iv) all transitive Noetherian linear frames, (v) any class of linear quasiorders. Then $Sat\mathcal{M}_{fin}(\mathcal{C})$ has BMP and is decidable. In particular decidable is $Sat\mathcal{M}_{fin}(\mathcal{C})$ for any $\mathcal{C} \in \{\mathcal{K}, \mathcal{S}5, \mathcal{S}4, \mathcal{KD}45, \mathcal{GL}.3, \mathcal{N}\}$ or $\mathcal{C} \subseteq \mathcal{S}4.3$.*

The proof of this result is different from those delivered above and can be found in the full paper.

6 POLYMODAL DESCRIPTION LOGICS

In order to extend the results obtained in the previous section to polymodal description logics, we show that the decidability of satisfiability is preserved under fusions of frame classes. More precisely, for classes $\mathcal{C}_1$ and $\mathcal{C}_2$ of frames of the form $\langle W_1, \lhd_1, \dots, \lhd_m \rangle$ and $\langle W_2, \lhd_{m+1}, \dots, \lhd_n \rangle$, respectively, the *fusion* $\mathcal{C}_1 \otimes \mathcal{C}_2$ of $\mathcal{C}_1$ and $\mathcal{C}_2$ consists of all frames $\langle W, \lhd_1 \dots, \lhd_n \rangle$ such that $\langle W, \lhd_1, \dots, \lhd_m \rangle \in \mathcal{C}_1$, $\langle W, \lhd_{m+1}, \dots, \lhd_n \rangle \in \mathcal{C}_2$. For example, $\mathcal{S}5_{n+1} = \mathcal{S}5_n \otimes \mathcal{S}5_1$, for any $n \geq 1$. By extending the technique of Kracht and Wolter (1991) developed for pure modal logics (see also (Fine and Schurz 1996), (Gabbay 1996), (Wolter 1997)), one can prove the following:

Theorem 26 *For any two classes of frames $\mathcal{C}_1$, $\mathcal{C}_2$,*

(i) *if $Sat\mathcal{M}(\mathcal{C}_i)$ is decidable, for $i = 1, 2$, then $Sat\mathcal{M}(\mathcal{C}_1 \otimes \mathcal{C}_2)$ is decidable;*

(ii) *if $Sat\mathcal{M}_{fin}(\mathcal{C}_i)$ is decidable, for $i = 1, 2$, then $Sat\mathcal{M}_{fin}(\mathcal{C}_1 \otimes \mathcal{C}_2)$ is decidable;*

(iii) *if $Sat\mathcal{M}_{fin}(\mathcal{C}_i) = Sat\mathcal{M}(\mathcal{C}_i)$, for $i = 1, 2$, then $Sat\mathcal{M}_{fin}(\mathcal{C}_1 \otimes \mathcal{C}_2) = Sat\mathcal{M}(\mathcal{C}_1 \otimes \mathcal{C}_2)$.*

For a proof we refer to the full paper. As a consequence we obtain

Theorem 27 *There exists a satisfiability checking algorithm for each of the following classes of models: $\mathcal{M}(\mathcal{K}_n)$, $\mathcal{M}(\mathcal{KD}45_n)$, $\mathcal{M}(\mathcal{S}5_n)$, $\mathcal{M}(\mathcal{GL}_n)$, $\mathcal{M}(\mathcal{S}4_n)$, where $n \geq 1$.*

7 CONCLUSION

In this paper we have shown the decidability of the satisfiability problem for most of the standard systems of epistemic and temporal description logics. It would be of interest, however, to analyze a number of more complex systems. For instance, in epistemic logic we did not touch on the common knowledge operator which is interpreted by the transitive and reflexive closure of the union of the accessibility relations for the individual agents. In the temporal case we studied only the simplest models with the operator "eventually". However, many applications require more expressive sets of modal operators (like "Next", "Until", "in the past", etc.). For those more complex systems first decidability results were obtained recently by the authors.

This paper provides rather general methods of establishing decidability of modal description logics. It does not analyze specific systems and the corresponding decision algorithms in detail. But this will certainly be necessary in order to make modal description logics applicable. An important task is to develop reasonably efficient algorithms for checking satisfiability and to determine the complexity of the satisfiability problem.

Our decision algorithms can treat models with varying, expanding and constant domains. But they are oriented to rigid designators, and it is not clear how to extend the algorithms to cover non-rigid ones.

Also we would like to draw attention to some interesting mathematical problems concerning modal description logics. For instance, what is the connection between the decidability (or the finite model property) of the pure modal logic determined by a class of frames $\mathcal{C}$ and the decidability of $Sat\mathcal{M}(\mathcal{C})$? How does the decidability depend on the cardinality of domains?

Acknowledgements

The work of the second author was supported by the Alexander von Humboldt Foundation and by grant 96–01–01205 from the Russian Fundamental Research Foundation. The authors are grateful to Franz Baader for many helpful remarks on the original version of the paper.

References

F. Baader, H.-J. Bürkert, J. Heinsohn, B. Hollunder, J Müller, B. Nebel, W. Nutt, and H.J. Profitlich (1990).. Terminological knowledge representation: a proposal for terminological logic. Technical Report TM-90-04, Deutsches Forchungszentrum für

Künstliche Intelligenz (DFKI).

F. Baader and B. Hollunder (1991). A terminological knowledge representation system with complete inference algorithms. In *Proceedings of the workshop on Processing Declarative Knowledge, PDK-91*, 67–86. Lecture Notes on Artificial Intelligence, No. 567. Springer Verlag.

F. Baader and A. Laux (1995). Terminological logics with modal operators. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 808–814, Montreal, Canada. Morgan Kaufman.

F. Baader and H.J. Ohlbach (1995). A multi-dimensional terminological knowledge representation language. *Journal of Applied Non-Classical Logic*, 5:153–197.

A. Borgida, R.J. Brachman, D.L McGuiness, and L. Alperin Resnick (1989). CLASSIC: A structural data model for objects. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 59–67. Portland, Oreg.

R.J. Brachman and J.G. Schmolze (1985). An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9:171–216.

D. Calvanese (1996). Finite model reasoning in description logics. In *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning*, 292–303. Morgan Kaufman, Los Altos.

A.V. Chagrov and M.V. Zakharyaschev (1997). *Modal Logic.* Clarendon Press, Oxford.

F. Donini, M. Lenzerini, D. Nardi, and A. Schaerf (1996). Reasoning in description logics. In G. Brewka, editor, *Principles of Knowledge Representation*, 191–236. CSLI Publications.

K. Fine and G. Schurz (1996). Transfer theorems for stratified modal logics. In J. Copeland, editor, *Logic and Reality, Essays in Pure and Applied Logic. In memory of Arthur Prior*, 169–213. Oxford University Press.

M. Finger and D. Gabbay (1992). Adding a temporal dimension to a logic system. *Journal of Logic, Language and Information*, 2:203–233.

M. Fitting (1993). Basic modal logic. In D.M. Gabbay, C.J. Hogger, and J.A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 1, 368–448. Oxford Scientific Publishers.

D.M. Gabbay (1996). Fibred semantics and the weaving of logics. Part 1: Modal and intuitionistic logics. *Journal of Symbolic Logic*, 61:1057–1120.

G. De Giacomo and M. Lenzerini (1994). Boosting the correspondence between description logics and propositional dynamic logics. In *Proceedings of the 12th Nat. Conf. on Artificial Intelligence (AAAI-94)*, 205–212. AAAI Press/The MIT Press.

J. Halpern and Yo. Moses (1992). A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54:319–379.

M. Kracht and F. Wolter (1991). Properties of independently axiomatizable bimodal logics. *Journal of Symbolic Logic*, 56:1469–1485.

S. Kripke (1980). *Naming and Necessity.* Harvard University Press.

A. Laux (1994). Beliefs in multi-agent worlds: a terminological approach. In *Proceedings of the 11th European Conference on Artificial Intelligence*, 299–303, Amsterdam.

P.F. Patel-Schneider (1989). Undecidability of subsumption in NIKL. *Artificial Intelligence Journal*, 39:263–272.

K. Schild (1993). Combining terminological logics with tense logic. In *Proceedings of the 6th Portuguese Conference on Artificial Intelligence*, pages 105–120, Porto.

M. Schmidt-Schauß (1989). Subsumption in KL-ONE is undecidable. In R.J. Brachman, H. Levesque, and R. Reiter, editors, *Proceedings of the 1st Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-89)*, 421–431. Morgan Kaufman, Los Altos.

E. Spaan (1993). *Complexity of Modal Logics.* PhD thesis, Department of Mathematics and Computer Science, University of Amsterdam.

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

F. Wolter (1998). Fusions of modal logics revisited. In M. Kracht, M. de Rijke, H. Wansing and M. Zakharyaschev, editors, *Advances in Modal Logic*, CSLI, Stanford.

Nested Graphs: A Graph-based Knowledge Representation Model with FOL Semantics

M. Chein
LIRMM (CNRS and Univ. Montpellier II)
161, rue Ada, F-34392 Montpellier Cedex 5
email: *name*@lirmm.fr

M.L. Mugnier
LIRMM

G. Simonet
LIRMM

Abstract

We present a graph-based KR model issued from Sowa's conceptual graphs but studied and developed with a specific approach. Formal objects are kinds of labelled graphs, which may be simple graphs or nested graphs. The fundamental notion for doing reasonings, called projection (or subsumption), is a kind of labelled graph morphism. Thus, we propose a *graphical* KR model, where "graphical " is used in the sense of [Sch91], i.e. a model that "uses graph-theoretic notions in an essential and nontrivial way". Indeed, morphism, which is the fundamental notion for any structure, is at the core of our theory. We define two first order logic semantics, which correspond to different intuitive semantics, and prove in both cases that projection is sound and complete with respect to deduction.

1 Introduction

Different kinds of *labelled graphs* have long been used for representing knowledge. In artificial intelligence, they have been investigated under the name of *semantic networks* (see for instance [Leh92] for an overview of recent research in this domain). [Sow76] proposed a model of this family, named *conceptual graphs* (CGs), which was developed in [Sow84]. The formal aspects of this model have two mathematical bases: logic and graph theory. Emphasizing one or the other leads to different research lines.

The former, a continuation of Peirce's work on existential graphs [Rob92], investigates CGs as a *diagrammatic* system of logic (for instance [Sow94] [vdB94]). The later develops CG as a *graphical* knowledge representation model, where "graphical" is used in the sense of [Sch91], i.e. a model that "uses graph-theoretic notions in an essential and nontrivial way" (for instance [LE91] [CM92]). Our work is concerned with this second viewpoint.

The basic notion is that of a *simple conceptual graph* (SG). A SG is a labelled bipartite graph where nodes of one class represent entities, nodes of the other class represent relations between these entities, and the labels (which are partially ordered) represent types and referents. The fundamental operation, called "projection", is a *morphism* between SGs. Projection induces a structural *subsumption* relation. SGs are provided with a semantics in first order logic (FOL), classically called Φ. Projection has been proven to be sound with respect to deduction [Sow84]. We proved that it is also complete, when SGs are in a "normal" form [CM92].

In this paper, we provide another FOL semantics, for which completeness holds without any restriction. We also present *nested conceptual graphs* (NGs), which are generalizations of SGs: in NGs, concept nodes may express complex information represented by NGs. We study NGs from a graphical viewpoint. More specifically:

- a NG is a combinatorial object recursively defined from SGs;
- the main notion, projection, is a morphism between NGs, from which a subsumption relation is defined;
- the NG model is provided with two logical semantics in FOL, which extend those for SGs. Soundness and completeness results are given for both cases.

NGs come from several applications in which the SG model proved to be inconvenient for expressing the knowledge involved. Let us give one example of an application we are currently involved in. In this application, NGs are used to represent and simulate the required behavior of human agents in case of emergency situations, for instance a fire in a chemical plant [BBV97]. This behavior is described by NGs which represent situations, pre-conditions and post-conditions of actions to be performed, each of these elements being themselves described by NGs. Knowledge is structured by hierarchical levels and reasonings must respect these levels.

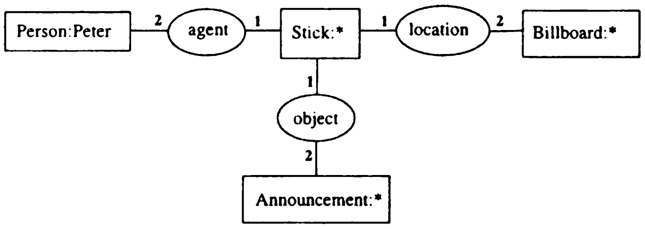
The paper is organized as follows. In section 2, basic definitions and results on SGs are provided. In section 3, co-reference links are added to SGs and we propose a new logical semantics, which is compared to Φ. We prove that projection is sound and complete w.r.t. this semantics. The model is then extended to NGs with co-reference links (section 4). Two logical semantics for NGs are developed, for which projection is proven to be sound and complete (sections 5 and 6). Section 7 is devoted to related works.

2 Basic Notions on Conceptual Graphs

In this section, we present main features about SGs. For a more detailed presentation see [Sow84] and [CM92]. Let us first specify some notations. Any partial order or preorder (reflexive and transitive but not necessarily antisymmetric relation) is denoted by $\leq$ or $\geq$. $\models$ is the logical consequence symbol in FOL.

Basic ontological knowledge is encoded in a structure we call a **support**. We consider here a simplified version of a support with 3 components, say $S = (T_C, T_R, I)$. T_C and T_R are type sets, respectively a set of concept types and a set of relation types, partially ordered by an A-Kind-Of relation. Relation types may have any arity greater or equal to 1, and two comparable relation types must have the same arity. I is a set of individual markers. All supports also possess a generic marker denoted by $*$. The following partial order is defined on the set of markers $I \cup \{*\}$: $*$ is the greatest element and elements of I are pairwise noncomparable.

Asserted facts are encoded by SGs. SGs are also used for representing ontological knowledge (such as type definitions) or other kinds of knowledge such as rules or prototypes, but we will not cover these aspects here. A **SG** related to a support S, is a labelled bipartite graph $G = (R, C, E, l)$. R and C are the node sets, respectively relation node set and concept node set. E is the set of edges. Edges incident on a relation node are totally ordered, they are numbered from 1 to the degree of the relation node. The *ith* neighbor of a relation r is denoted by $G_i(r)$. Each node has



This graph may represent the information "Peter sticks an announcement on a billboard".

Figure 1: A simple graph

a label given by the mapping l. A concept node c is labelled by a couple $(type(c), ref(c))$, where $type(c)$, the type of c, belongs to T_C, and $ref(c)$, the referent of c, either belongs to I — then c is said to be an individual concept node — or is the generic marker $*$ — then c is said to be a generic concept node. A relation node r is labelled by a relation type, $type(r)$, and the number of edges incident on r is equal to the arity of $type(r)$. Figure 1 gives an example of a SG.

SGs are given a semantics in FOL, denoted by Φ [Sow84]. Given a support S, a constant is assigned to each individual marker and an n-adic (resp. unary) predicate is assigned to each n-adic relation (resp. concept) type. For simplicity, we consider that each constant or predicate has the same name as the associated element of the support. To S is assigned a set of formulas, $\Phi(S)$, which corresponds to the interpretation of the partial orderings of T_R and T_C. For all types t_1 and t_2 such that $t_1 \geq t_2$, one has the formula $\forall x_1 ... x_p (t_2(x_1, ..., x_p) \rightarrow t_1(x_1, ..., x_p))$, where $p = 1$ for concept types, and p is otherwise the arity of the relation. Φ maps any graph G on S into a formula $\Phi(G)$ in the following way. First assign to each concept node c a term which is a variable if c is generic, and otherwise the constant corresponding to $ref(c)$. Two distinct generic nodes receive distinct variables. Then assign an atom to each node of G: the atom $t_c(e)$ to a concept c where t_c stands for the type of c and e is the term associated to c; the atom $t_r(e_1, ..., e_p)$ to a relation r where t_r stands for the type of r and e_i is the term associated to the *ith* neighbor of r. Finally, the conjunction of these atoms is made and the obtained formula is existentially closed. E.g. let us consider the graph G of Figure 1. We call c_1, c_2, c_3, c_4 the concept nodes of respective types *Person*, *Stick*, *Announcement*, *Billboard* and assign the variable y_i to the generic concept node c_i for $i = 2, 3, 4$. The formula associated to G is
$\Phi(G) = \exists y_2 y_3 y_4 (Person(Peter) \wedge Stick(y_2) \wedge Announcement(y_3) \wedge Billboard(y_4) \wedge$

$agent(y_2, Peter) \wedge object(y_2, y_3) \wedge location(y_2, y_4))$.

Several graph operations are defined on SGs for doing reasonings. Our work is based on the projection operation [Sow84]. A **projection** from a SG $G = (R_G, C_G, E_G, l_G)$ to a SG $H = (R_H, C_H, E_H, l_H)$ is a mapping Π from R_G to R_H and from C_G to C_H which

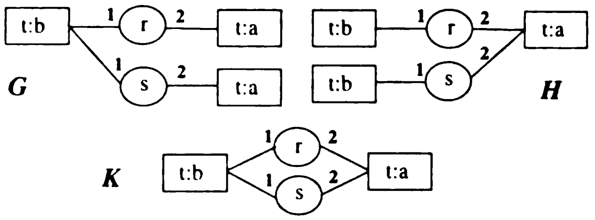
1. preserves adjacency and order on edges: $\forall\ rc \in E_G\ \Pi(r)\Pi(c) \in E_H$ and if $c = G_i(r)$ then $\Pi(c) = H_i(\Pi(r))$
2. may decrease labels: $\forall x \in R_G \cup C_G\ l_G(x) \geq l_H(\Pi(x))$.

The partial order on relation labels is that of T_R. The partial order on concept labels is the product of the partial order on T_C and the partial order on $I \cup \{*\}$. Note that projection is generally not a one-to-one mapping ($\Pi(G)$ is generally not isomorphic to G). See for instance Figure 2: G and H project onto K.

Projection can also be defined as a graph (homo-) morphism. Let us recall that a graph morphism from a graph G to a graph H is a mapping f from G node set to H node set which preserves adjacency, i.e. if xy is an edge of G then $f(x)f(y)$ is an edge of H. When G and H are labelled, morphism may satisfy additional constraints on labels. Projection is a labelled bipartite graph morphism.

Projection defines a **subsumption** relation on SGs: $G \geq H$ (G subsumes H) if there is a projection from G to H. This relation is a preorder, but not an order. Subsumption checking is an NP-complete problem [CM92]. It becomes polynomial when G is a tree [Mug95] (or more generally when G has a bounded treewidth). Also note that there is a "strong" equivalence between the subsumption checking problem and the constraint satisfaction problem [Dec92], allowing direct transfer of efficient algorithms from one problem to the other [MC96].

Projection is sound and complete with respect to FOL, i.e.: given two SGs G and H defined on S, if G subsumes H then $\Phi(S), \Phi(H) \models \Phi(G)$ (soundness [Sow84]); conversely, if H is in normal form and $\Phi(S), \Phi(H) \models \Phi(G)$ then G subsumes H (completeness [CM92]). A SG is said to be in **normal form** if each individual marker appears at most once in concept node labels. A SG G can be put into normal form by identifying all concept nodes having the same individual referent, provided that types of these nodes possess a greatest lower bound (which is the type of the obtained node). This simple transformation produces a SG - called *the normal form* of G - whose



$\Phi(G) = t(b) \wedge t(a) \wedge t(a) \wedge r(b,\ a) \wedge s(b,\ a)$.
$\Phi(H) = t(b) \wedge t(b) \wedge t(a) \wedge r(b,\ a) \wedge s(b,\ a)$.
$\Phi(G)$ and $\Phi(H)$ are equivalent, while G and H are noncomparable by projection. K is the normal form of G and H. G and H project into K.

Figure 2: Normal Form

logical semantics is trivially equivalent to that of G. See Figure 2.

Let add that there are strong links between conjunctive query containment in databases and projection. Indeed, it is straightforward to represent any positive and non recursive conjunctive query q by a simple graph G_q in such a way that, given two queries q and q', q' contains q if and only if there is a projection from G_q to $G_{q'}$. In this transformation, the obtained support for SGs is very simple, since all types are noncomparable. More precisely, let us consider a positive and non recursive conjunctive query $q = ans(u) \leftarrow r_1(u_1),\ ...\ r_n(u_n),\ n \geq 1$, where, given R a set of relation names and *dom* a set of constants, $r_1\ ...\ r_n$ belong to R, *ans* is not in R, u and $u_1\ ...\ u_n$ are tuples of terms (variables or constants of *dom*), and each variable of u occurs at least once in $u_1\ ...\ u_n$. We follow here the notations of [AHV95]. Now, let $S = (T_C,\ T_R,\ I)$ be a support built as follows. T_C is restricted to a single type. T_R is in bijection with $R \cup Ans$, where $Ans = \{ans_i |\ i \geq 1\}$ and i stands for the arity of *ans*. I is in bijection with *dom*. Elements of T_R are noncomparable. Then a (normal) simple graph $G_q = (R_q, C_q, E_q, l_q)$ can be assigned to q as follows. R_q is a set of (n+1) nodes labelled by $r_1,\ ...,\ r_n,\ ans$. C_q is in bijection with the set of terms occurring in $u_1,\ ...,\ u_n$. And if t is the *ith* argument of r_j (or *ans*), then the concept node assigned to t is the *ith* neighbor of the relation node assigned to r_j (or *ans*). Then, one can immediately check that, given two queries, q and q', there is a homomorphism from q to q' if and only if there is a projection from G_q to $G_{q'}$.

Moreover, it can be proven that the homomorphism theorem ([AHV95], 6.2.3) is equivalent to the soundness and completeness theorem for projection between (normal) SGs [CM92], the minimization theorem ([AHV95], 6.2.6) is equivalent to the theorem about irredundant SGs [CM92], and the complexity theorem for query decision problems ([AHV95], 6.2.10) is equivalent to the complexity theorem for SGs decision problems [CM92]. See [SCM98] for more details about these equivalences.

3 Co-reference links and another FOL semantics

According to the semantics Φ, merging several individual concept nodes having the same referent (i.e. representing the same entity) does not change the meaning of the graph. However, in some applications, it may be useful to represent the same entity with different concept nodes corresponding to different aspects or viewpoints on this entity. For instance, suppose G represents the organization of a company. G contains three subgraphs, which respectively represent the structures of the staff, the joint production committee and the football team. Some persons appear in more than one of these three structures, but we do not wish to identify the concept nodes representing the same person, which would have the effect of mixing and destroying the three structures. When a specified individual is represented by several nodes, the individual marker suffices for expressing that these nodes represent the same entity. But in the case of an unspecified entity represented by several generic nodes (e.g. a person identified by his/her different functions in the company) an additional structure, called a co-reference link, is needed.

A SG with co-reference links (in short $\mathbf{SG^{ref}}$) is a graph $G = (R, C, E, l, co\text{-}ref)$, where **co-ref** is an equivalence relation on the set of G generic concept nodes (the *co-reference* relation).The intuitive semantics of $co\text{-}ref$ is "represents the same entity as". Any basic SG $G = (R, C, E, l)$ may be written in the form $G = (R, C, E, l, co\text{-}ref)$ where $co\text{-}ref$ is restricted to the identity relation. The $co\text{-}ref$ relation is naturally extended to an equivalence relation **co-ident** on the set C of all concept nodes of G (the *co-identity* relation). Every equivalence class for the co-identity relation is a set of concept nodes representing the same entity which are either co-referent generic nodes (in that case they are explicitly linked by a co-reference link in a graphical representation of G) or individual nodes having the same referent: $\forall c, c' \in C$,
$co\text{-}ident(c, c')$ iff $(co\text{-}ref(c, c')$ or $ref(c) = ref(c') \in I)$

The definitions and results on SGs are modified as follows by the introduction of co-reference links. A **projection** from a SG^{ref} $G = (R_G, C_G, E_G, l_G, co\text{-}ref_G)$ to a SG^{ref} $H = (R_H, C_H, E_H, l_H, co\text{-}ref_H)$ is a projection Π from the SG (R_G, C_G, E_G, l_G) to the SG (R_H, C_H, E_H, l_H) that preserves co-identity, i.e. $\forall\ c, c' \in C_G$ if $co\text{-}ref_G(c, c')$ then $co\text{-}ident_H(\Pi(c), \Pi(c'))$. Φ may be extended to SG^{ref}s by associating the same variable to co-referent generic concept nodes. A SG^{ref} G is in normal form iff the co-identity relation on C_G is the identity one (i.e. iff it contains no co-reference links and each individual marker appears at most once in concept node labels). A SG^{ref} is put into normal form by identifying its co-identical nodes. In order to translate the whole information encoded in a graph, we introduce another FOL semantics, called Ψ. In this semantics, two terms are assigned to a concept node c. The first term $(e_{\bar{c}})$ represents the co-identity class $\bar{c}$ of c as in Φ (it is a constant if c is an individual node, otherwise it is the variable associated with $\bar{c}$). The second term (e_c) is a variable, which represents the node itself. All these variables are pairwise distinct. $\Psi(S)$ is defined as $\Phi(S)$, except that predicates associated with concept types become binary. $\Psi(G)$ is defined as follows. Assign the atom $t(e_{\bar{c}}, e_c)$ to each concept node c and the atom $t_r(e_{c_1}, \ldots, e_{c_p})$ to each relation node r of G, where c_i is the ith neighbor of r. $\Psi(G)$ is the existential closure of the conjunction of these atoms. E.g. the formula associated to the graph G of Figure 2 is $\Psi(G) = \exists x_1 x_2 x_3 (t(b, x_1) \wedge t(a, x_2) \wedge t(a, x_3) \wedge r(x_1, x_2) \wedge s(x_1, x_3))$. With the graph G of Figure 1, we assign the variable y_i to the co-reference class restricted to the generic concept node c_i for $i = 2, 3, 4$ (as G does not contain any co-reference link) and the variable x_i to the concept node c_i for $i = 1, 2, 3, 4$. We obtain $\Psi(G) = \exists y_2 \ldots y_4 x_1 \ldots x_4 (Person(Peter, x_1) \wedge Stick(y_2, x_2) \wedge Announcement(y_3, x_3) \wedge Billboard(y_4, x_4) \wedge agent(x_2, x_1) \wedge object(x_2, x_3) \wedge location(x_2, x_4))$.

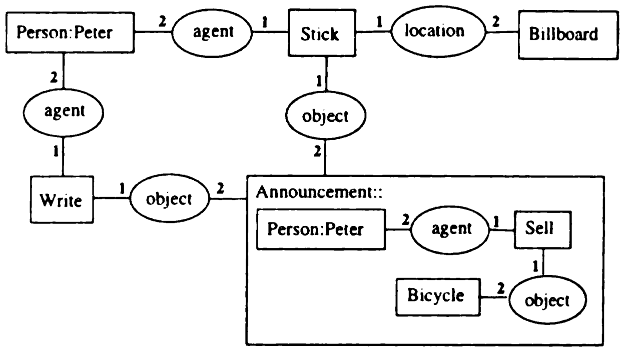
Projection is sound and complete with respect to this semantics without any restriction. E.g. consider the graphs of Figure 2. $\Psi(G)$ and $\Psi(H)$ are noncomparable by deduction and are both deducible from $\Psi(K)$.

Theorem 1 *Let G and H be two SG^{ref}s.*
G subsumes H iff $\Psi(S), \Psi(H) \models \Psi(G)$.

Proof: the proof is similar to that with the semantics Φ. However we give it here since the complete proof with Φ is available in French only [MC96]. For any formula F, let $\mathcal{C}(F)$ denote the clausal form of F. $\mathcal{C}(\Psi(H))$ is the set of atomic clauses obtained from $\Psi(H)$ atoms by substituting Skolem constants to the variables of $\Psi(H)$. Let ρ be this substitution. $\mathcal{C}(\neg\Psi(G))$ contains a unique clause whose litterals are the negations of $\Psi(G)$ atoms. The Herbrand Universe U_H of the formula $\Psi(S) \wedge \Psi(H) \wedge \neg\Psi(G)$ is the set of constants appearing in $\mathcal{C}(\Psi(H))$ or in $\Psi(G)$. A *s-substitution* from G to H (s standing for *subsuming*) is a substitution σ of the variables of $\Psi(G)$ by U_H constants such that for any atom $t(e_1, ..., e_n)$ of $\Psi(G)$, there is a t' such that $t' \leq t$ and $\sigma(t'(e_1, ..., e_n))$ is an atom of $\mathcal{C}(\Psi(H))$. Theorem 1 immediately follows from Lemmas 1 and 2. □

Lemma 1: *$\Psi(S), \Psi(H) \models \Psi(G)$ iff there is a s-substitution from G to H.*
Proof: let $\mathcal{C} = \mathcal{C}(\Psi(S) \wedge \Psi(H) \wedge \neg\Psi(G))$. $\Psi(S), \Psi(H) \models \Psi(G)$ iff $\mathcal{C}$ is unsatisfiable. If there is a s-substitution from G to H then the empty clause can be obtained from $\mathcal{C}$ by the resolution method, so $\mathcal{C}$ is unsatisfiable. Conversely, let us suppose that $\mathcal{C}$ is unsatisfiable. Let v be the Herbrand interpretation defined by: for any predicate t of arity n and for all constants a_1, ..., a_n of U_H, $v(t)(a_1, ..., a_n)$ is true iff there is a $t' \leq t$ such that $t'(a_1, ..., a_n)$ is an atom of $\mathcal{C}(\Psi(H))$. v is a model of $\mathcal{C}(\Psi(S) \wedge \Psi(H))$. Then v is not a model of $\mathcal{C}(\neg\Psi(G))$, which provides a s-substitution from G to H. □

Lemma 2: *G subsumes H iff there is a s-substitution from G to H.*
Proof: let us suppose that G subsumes H. Let Π be a projection from G to H. For any variable x of $\Psi(G)$, let c be the concept node of G such that $x = e_c$ (resp. $e_{\overline{c}}$) and let $\sigma(x)$ be the U_H constant $\rho(e_{\Pi(c)})$ (resp. $\rho(e_{\overline{\Pi(c)}})$). For any atom $t(e_1, ..., e_n)$ of $\Psi(G)$, let s be the node of G associated to this atom. The atom of $\mathcal{C}(\Psi(H))$ associated to $\Pi(s)$ is in the form $\sigma(t'(e_1, ..., e_n))$, with $t' \leq t$. σ is a s-substitution from G to H.
Conversely, let us suppose that there is a s-substitution σ from G to H. For any node s of G, let $t(e_1, ..., e_n)$ be the atom of $\Psi(G)$ associated to s and let $\Pi(s)$ be a node of H associated to an atom in the form $\sigma(t'(e_1, ..., e_n))$, with $t' \leq t$. Π fullfills condition 2 of the projection definition. As for condition 1, if $c = G_i(r)$ then $\sigma(e_c) = \rho(e_{\Pi(c)}) = \rho(e_{H_i(\Pi(r))})$, then $\Pi(c) = H_i(\Pi(r))$. Note that with the semantics Φ, we only have $\sigma(e_{\overline{c}}) = \rho(e_{\overline{\Pi(c)}}) = \rho(e_{\overline{H_i(\Pi(r))}})$, then $\overline{\Pi(c)} = \overline{H_i(\Pi(r))}$ and we need the normality condition to ensure that $\Pi(c) = H_i(\Pi(r))$. Π is a projection from G to H. □



(* and ** are omitted)

Figure 3: A nested graph

4 Nested Graphs

In this section, we define nested graphs (without co-reference links first) and projection between nested graphs.

Let us consider the graph of Figure 1. Suppose one wants to add two pieces of information about the announcement. First, "Peter wrote this announcement", second "this announcement says that Peter is selling a bicycle". Let us assume that these pieces of information are considered to be of different levels, the first one being an *external* information about the announcement, and the second one an *internal* information. If one wants to keep this difference in the representation, then in the former the announcement node is a *black box* which is joined to the Peter node via a node representing a writing, and the latter information can be put inside the announcement node. This last information can be obtained by zooming on the announcement node which in this case becomes a *glass box* (Figure 3). It can also be said that the information nested in a node is pertinent within the context represented by this node. This information is called a *(partial) description* of the node.

Note that a description is not a type definition. First, a description applies to a specific concept node, whereas a graph defining a type applies to all concept nodes of this type. Second, a type definition provides a characterization of a type, i.e. it describes necessary and sufficient conditions for any individual to belong to the type, whereas a description is only a partial information about an individual. Two co-identical nodes may have different descriptions, even in the same context.

NGs are defined by structural induction as follows.

1. A *basic NG* is obtained from a SG by adding to the label of each concept node c, a third field, denoted

$Desc(c)$, equal to ** (** can be considered as the empty description, and a trivial bijection exists between basic NGs and SGs);

2. Let G be a basic NG, let c_1, c_2, ..., c_k be concept nodes of G, and let G_1, G_2, ..., G_k be NGs. The graph obtained by substituting G_i to the description ** of c_i for $i = 1$, ..., k is a NG.

It is important to note (for following definitions of $A(G)$ and UC_G) that if a SG or a NG H is used several times in the construction of a NG G, we consider that several copies of H (and not several times the graph H itself) are used in the construction of G.

Any NG G has an associated syntactic tree $\mathbf{A(G)}$ whose nodes are SGs appearing in the construction of G. More precisely, $A(G)$ is a rooted tree which can be defined by structural induction as follows, with $R(G)$ denoting the root of $A(G)$: if G is the basic NG obtained from the SG G' then $A(G)$ is restricted to the node G'; if G is the NG obtained from the basic NG H, concept nodes c_1, c_2, ..., c_k, and NGs H_1, ..., H_k, then $A(G)$ is built from $A(H_1)$, ..., $A(H_k)$ by adding $R(H)$ as the root and edges $(R(H), c_i)R(H_i)$ leading from the node c_i of H to the root of $A(H_i)$. Note that for any NG G, $(J, c)K$ is an edge of $A(G)$ iff J and K are nodes of $A(G)$, c is a concept node of J and K is the root of the tree associated to the description graph of c (i.e. $K = R(Desc(c))$).

A **NG** can be denoted by $G = (R, C, E, l)$ where R, C and E are respectively relation, concept, and edge sets of the SG $R(G)$[1], l is a labelling function of R and C such that $\forall r \in R$, $l(r) \in T_R$ and $\forall c \in C$, $l(c) = (type(c), ref(c), Desc(c))$ with $type(c) \in T_C$, $ref(c) = *$ or $ref(c) \in I$, $Desc(c) = **$ or $Desc(c)$ is a NG. A *complex concept node* is a node with a nonempty (i.e. different from **) $Desc$. The set of complex nodes of G is denoted by $D(G)$.

We distinguish the set C_G of concept nodes of a NG G from the set UC_G of concept nodes ***appearing in*** G, i.e. the set of concept nodes of all SGs used in the construction of G. The set UC_G is defined by induction on the depth of G (the depth of G is that of the rooted tree $A(G)$, i.e. the maximum length of a path in $A(G)$). A projection from a NG G to a NG H is defined by induction on the depth of G. A **projection** from a NG $G = (R_G, C_G, E_G, l_G)$ to a NG $H = (R_H, C_H, E_H, l_H)$ is a family of mappings $\Pi = \Pi_0, \Pi_{c_1} \dots \Pi_{c_k}$, with $D(G) = \{c_1 \dots c_k\}$, which satisfies:

1. Π_0 is a projection from the SG $R(G)$ to the SG $R(H)$,
2. $\forall c_i \in D(G)$, $\Pi_0(c_i) \in D(H)$ and Π_{c_i} is a projection from $Desc(c_i)$ to $Desc(\Pi_0(c_i))$.

A projection Π from a NG G to a NG H naturally induces a mapping (also noted Π for simplicity) from UC_G to UC_H defined by induction on the depth of G. For any c in UC_G, if c is in C_G then $\Pi(c)$ is equal to $\Pi_0(c)$ and otherwise $\Pi(c)$ is equal to $\Pi_{c_i}(c)$, where c_i is the node of $D(G)$ such that c is in $UC_{Desc(c_i)}$.

This notion of a projection can be formulated in terms of the rooted trees associated to NGs. Let $A_1 = (X_1, U_1)$ and $A_2 = (X_2, U_2)$ be two rooted trees of SGs with respective roots R_1 and R_2. An **A-projection** from A_1 to A_2 is a family of mappings $\varphi = \varphi_0, \varphi_{K_1} \dots \varphi_{K_p}$, with $X_1 = \{K_1 \dots K_p\}$, which satisfies:
1. φ_0 is a mapping from X_1 to X_2 such that $\varphi_0(R_1) = R_2$,
2. $\forall K \in X_1$, φ_K is a projection from K to $\varphi_0(K)$,
3. $\forall (J, c)K \in U_1$, $(\varphi_0(J), \varphi_J(c))\varphi_0(K) \in U_2$.
For all NGs G and H, there is a projection from G to H iff there is an A-projection from $A(G)$ to $A(H)$ [Sim96b].

As in SGs, several individual concept nodes with the same marker may appear in a NG, and it may be useful to express that several generic concept nodes represent the same entity. Note that such nodes may be identified if they appear in the same SG of $A(G)$ but not otherwise. Thus, the extension from SGs to NGs makes co-reference links necessary even in applications where the possibility of identifying nodes representing the same entity made co-reference links useless in SGs. A NG with co-reference links (in short $\mathbf{NG^{ref}}$) is a graph $G = (R, C, E, l, \textit{co-ref})$, where *co-ref* is an equivalence relation on the set of generic concept nodes appearing in G. The *co-ref* relation is extended to an equivalence relation *co-ident* on the set UC_G of concept nodes appearing in G as in the SG^{ref} model. A **projection** Π from a NG^{ref} G to a NG^{ref} H must preserve co-identity: $\forall\ c, c' \in UC_G$ if $\textit{co-ref}_G(c, c')$ then $\textit{co-ident}_H(\Pi(c), \Pi(c'))$.

Projection checking between two nested graphs, with or without co-reference links, is an NP-complete problem. For NGs (i.e. without co-reference links) the complexity of this problem is polynomially related to the complexity of projection checking for the class of simple graphs which make up the NGs. For instance,

[1]More precisely, R, C and E may be identified to relation, concept, and edge sets of the SG $R(G)$ respectively, as long as the description field in the concept node labels is not concerned.

it is polynomial when these graphs are trees.

5 The Semantics Φ

In order to extend Φ to NG^{ref}s, we add one argument to each predicate, called the **context argument**. Thus, a binary predicate is associated to each concept type, and a $(n+1)$-adic predicate is associated to each n-adic relation type. For instance, if x (resp. y, z) is the variable assigned to the generic concept node of type *Announcement* (resp. *Sell*, *Bicycle*), then the interpretation of the atom $obj(x,y,z)$ is: the bicycle z is the object of the sale y *in the context of* the announcement x. With S is associated the set $\Phi(S)$ of formulas $\forall z \forall x_1 \ldots x_p(t_2(z,x_1,\ldots,x_p) \rightarrow t_1(z,x_1,\ldots,x_p))$, where t_1 and t_2 are types such that $t_1 \geq t_2$ and p is the arity of t_1 and t_2 in the semantics Φ for SGs. For any NG^{ref} G, let $r(G)$ be the number of equivalence classes of the relation $co\text{-}ref_G$. Assign $r(G)$ distinct variables $y_1, \ldots, y_{r(G)}$ to the $co\text{-}ref_G$ classes. For any node K of $A(G)$, if K contains generic nodes then the variables of $\Phi(K)$ are some of the variables $y_1, \ldots, y_{r(G)}$. A formula $\Phi'(t,G')$ is associated with any term t and any subgraph G' of G (i.e. NG^{ref} G' either equal to G or to the description graph of a concept node appearing in G). $\Phi'(t,G')$ is the formula associated to G' when G' is in the context represented by the term t, or more specifically, when G' is the description graph of a concept node associated to the term t. $\Phi'(t,G')$ is defined by induction on the depth of G'. For any node K of $A(G)$, $\Phi'(t,K)$ denotes the conjunction of the atoms obtained from those of $\Phi(K)$ by adding the first argument t; for any concept node c of K, $e_{\bar{c}}$ denotes the term assigned to c in $\Phi(K)$ (which is in fact the term assigned to the co-identity class $\bar{c}$ of c).

$$\Phi'(t,G') = \Phi'(t,R(G')) \wedge (\wedge_{c \in D(G')} \Phi'(e_{\bar{c}}, Desc(c)))$$

The formula

$$\mathbf{\Phi(G)} = \exists y_1 \ldots y_{r(G)} \Phi'(a_0, G)$$

is associated to any NG^{ref} G defined on the support S, where a_0 is a constant representing the general context induced by the support S, so that the same constant a_0 is used for all NG^{ref}s defined on the support S. E.g. the formula associated to the graph G of Figure 3 is

$\Phi(G) = \exists y_2 \ldots y_5 y_7 y_8 (Person(a_0, Peter) \wedge Stick(a_0, y_2) \wedge Announcement(a_0, y_3) \wedge Billboard(a_0, y_4) \wedge Write(a_0, y_5) \wedge agent(a_0, y_2, Peter) \wedge object(a_0, y_2, y_3) \wedge location(a_0, y_2, y_4) \wedge agent(a_0, y_5, Peter) \wedge object(a_0, y_5, y_3) \wedge \Phi'(y_3, Desc(c_3)))$,

where c_3 is the concept node of type *Announcement* and

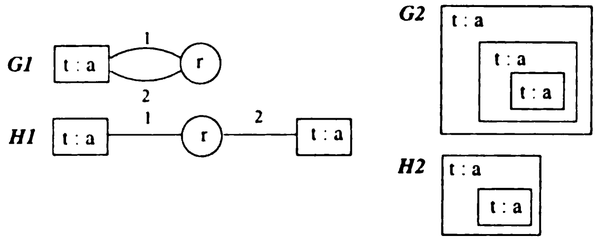


Figure 4: Counterexamples to the completeness result in NG^{ref}s with Φ

$\Phi'(y_3, Desc(c_3)) = Person(y_3, Peter) \wedge Sell(y_3, y_7) \wedge Bicycle(y_3, y_8) \wedge agent(y_3, y_7, Peter) \wedge object(y_3, y_7, y_8)$.

$\Phi(G)$ may also be defined from $A(G)$.

$$\Phi(G) = \exists y_1 \ldots y_{r(G)} (\wedge_{K \text{ node of } A(G)} \Phi'(t_K, K))$$

where $t_K = a_0$ if $K = R(G)$ and otherwise let $(J,c)K$ be the edge of $A(G)$ into K, $t_K = e_{\bar{c}}$.

The normality notion and the soundness and completeness result extend to NG^{ref}s. Let us first consider two counterexamples to the completeness result (presented in Figure 4). $\Phi(G_1) = t(a_0,a) \wedge r(a_0,a,\ a)$ and $\Phi(H_1) = t(a_0,a) \wedge t(a_0,a) \wedge r(a_0,a,\ a)$. $\Phi(G_1)$ and $\Phi(H_1)$ are equivalent formulas, but there does not exist a projection from G_1 to H_1. The trouble here is that H_1 is not a SG^{ref} in normal form. $\Phi(G_2) = t(a_0,a) \wedge t(a,\ a) \wedge t(a,a)$ and $\Phi(H_2) = t(a_0,a) \wedge t(a,a)$. The trouble here is that there are two co-identical concept nodes appearing in H_2 such that one is a complex node and the other is not.

These remarks give some intuition for extending normality to NG^{ref}s. Let G be a NG^{ref} and G' and H' be two SG^{ref}s nodes of $A(G)$ (resp. two NG^{ref}s subgraphs of G). An **exact projection** from G' to H' is a projection from G' to H' mapping each generic node c of $C_{G'}$ (resp. $UC_{G'}$) to a generic node of $C_{H'}$ (resp. $UC_{H'}$) co-referent to c in G. G' and H' are **exactly equivalent** iff there is an exact projection from G' to H' and from H' to G'. A NG^{ref} G is in **normal form** iff

1. every node of $A(G)$ is a SG^{ref} in normal form,

2. for all co-identical concept nodes c and c' appearing in two distinct nodes of $A(G)$, if c is a complex node then c' is a complex node and $R(Desc(c'))$ is exactly equivalent to $R(Desc(c))$.

Note that for all co-identical complex nodes c and c' appearing a NG^{ref} G in normal form, not only the SG^{ref}s $R(Desc(c))$ and $R(Desc(c'))$ are exactly equivalent, but also the whole NG^{ref}s $Desc(c)$ and $Desc(c')$. However, putting a NG^{ref} G into normal form (i.e. transforming G into a NG^{ref} G' in normal form such that $\Phi(G) \equiv \Phi(G')$) is not always possible. E.g. consider the graph H_2 of Figure 4. If H' were a NG^{ref} in normal form such that $\Phi(H_2) \equiv \Phi(H')$ then $\Phi(H')$ would contain the atom $t(a, a)$, then any concept node of $UC_{H'}$ with referent a would have a concept node with referent a in its description graph, and there would be an infinite chain in $A(H')$, which is impossible. The definition of normality may be weakened into that of k-normality in such a way that any NG^{ref} G may be put into k-normal form for any $k \geq depth(G)$. The *level* in G of a node c of UC_G is the level in $A(G)$ of the node K of $A(G)$ containing c (i.e. the number of edges of the path in $A(G)$ from $R(G)$ to K). A NG^{ref} G is in **k-normal form** iff

1. every node of $A(G)$ is a SG^{ref} in normal form,

2. for all co-identical concept nodes c and c' appearing in two distinct nodes of $A(G)$ such that c is a complex node, if the level of c' in G is less than k then c' is a complex node and if c' is a complex node then $R(Desc(c'))$ is exactly equivalent to $R(Desc(c))$.

Note that if G is in normal form then G is in k-normal form for any natural integer k.

Theorem 2 *Let G and H be two NG^{ref}s and let k be an integer such that $k \geq depth(G)$.*
If G subsumes H then $\Phi(S), \Phi(H) \models \Phi(G)$. If H is in k-normal form and $\Phi(S), \Phi(H) \models \Phi(G)$ then G subsumes H.

Sketch of proof: (the detailed proof may be found in [Sim96b]). Note that Lemmas 1 and 2 are true for NG^{ref}s instead of SG^{ref}s and the semantics Φ instead of Ψ (provided that in Lemma 2 H is in normal form). And Lemma 1 is true for NG^{ref}s instead of SG^{ref}s. Let us suppose that G subsumes H. Let $\varphi = \varphi_0, \varphi_{K_1} \dots \varphi_{K_p}$ be an A-projection from G to K. From Lemma 2, for any node K of $A(G)$, there is a s-substitution σ_K from K to $\varphi_0(K)$. Let σ be the substitution of the variabes of $\Phi(G)$ obtained from the σ_K. We prove that σ is a s-substitution from G to H and conclude with Lemma 1.
Conversely, let us suppose that H is in k-normal form with $k \geq depth(G)$ and $\Phi(S), \Phi(H) \models \Phi(G)$. From Lemma 1, there is a s-substitution σ from G to H. We construct an A-projection φ from $A(G)$ to $A(H)$. We define $\varphi_0(K)$ and φ_K for any node K of $A(G)$ by induction on the level l of K in $A(G)$. For $l = 0$, K is $R(G)$. The atoms of $\Phi(G)$ (resp $\mathcal{C}(\Phi(H))$) assigned to the nodes of $R(G)$ (resp. $R(H)$) are those with a_0 as first argument. Then the restriction of σ to the variables of $\Phi(R(G))$ is a s-substitution from $R(G)$ to $R(H)$. From Lemma 2, there is a projection Π_0 from $R(G)$ to $R(H)$. We define $\varphi_0(R(G)) = R(H)$ and $\varphi_{R(G)} = \Pi_0$. Suppose φ is defined up to level l. Let $(J, c)K$ be an edge of $A(G)$ with K at level $l + 1$. Let e_1 be the first argument of the atoms of $\Phi(G)$ assigned to the nodes of K. Any atom of $\mathcal{C}(\Phi(H))$ with $\sigma(e_1)$ as first argument is associated with a node of a SG^{ref} K' of $A(H)$, with $K' = R(Desc(c'))$ and c' is co-identical to $\varphi_J(c)$. Then, as H is in k-normal form and $level_H(\varphi_J(c)) = level_G(c) < depth(G) \leq k$, $\varphi_J^k(c)$ is a complex node and K' is exactly equivalent to $R(Desc(\varphi_J(c)))$. From Lemma 3 below, the restriction of σ to the variables of $\Phi(K)$ is a s-substitution from K to $R(Desc(\varphi_J(c)))$. From Lemma 2, there is a projection Π_K from K to $R(Desc(\varphi_J(c)))$. We define $\varphi_0(K) = R(Desc(\varphi_J(c)))$ and $\varphi_K = \Pi_K$. φ is an A-projection from $A(G)$ to $A(H)$, then G subsumes H □

Lemma 3: *For any NG^{ref} H, for all exactly equivalent nodes K and K' of $A(H)$ and for any atom $t(a_1, ..., a_n)$, there is $t' \leq t$ such that $t'(a_1, ..., a_n)$ is an atom of $\mathcal{C}(\Phi(H))$ assigned to a node of K iff there is $t' \leq t$ such that $t'(a_1, ..., a_n)$ is an atom of $\mathcal{C}(\Phi(H))$ assigned to a node of K'.*
Proof: the result follows from the fact that if s is a node of K associated to the atom $t'(a_1, ..., a_n)$ of $\mathcal{C}(\Phi(H))$ and Π is an exact projection from K to K', then the atom of $\mathcal{C}(\Phi(H))$ associated to the node $\Pi(s)$ of K' is in the form $t''(a_1, ..., a_n)$, with $t'' \leq t'$. □

Any NG^{ref} G may be put into k-normal form for any $k \geq depth(G)$ as follows. A rooted tree A_k of SG^{ref}s is built level by level from its root to level k. Its root is a copy of $R(G)$. If J is a node of A_k at level $l < k$, and if c is a concept node of J (c is a copy of a node c' of UC_G) and if there is at least one complex node co-identical to c' in G, then add the edge $(J, c)K$ to A_k, where K is a copy of the union of the SG^{ref}s $R(Desc(c''))$ for any complex node c'' co-identical to c' in G. Two generic nodes appearing in nodes of A_k are co-referent iff they are copies of co-referent nodes of UC_G. Let G'_k be the NG^{ref} such that $A(G'_k)$ is obtained from A_k by putting each node of A_k into normal form (when identifying several co-identical concept nodes, only one of the subtrees issued from these nodes is kept, as these subtrees are exact copies of each other in A_k). G'_k is in k-normal form and $\Phi(G) \equiv \Phi(G'_k)$. G'_k is called *the k-normal form* of G. E.g. the 2-normal form of the graph H_2 of Figure 4 is H_2 and its 3-normal form is G_2.

The semantics Φ is unable to express not only that an entity is represented by several concept nodes in a node of $A(G)$, but also that several concept nodes representing the same entity have different descriptions. Φ may be pertinent in applications in which the meaning of a graph is not changed when identifying co-identical concept nodes of a SG^{ref} or replacing the description of co-identical concept nodes by the union of their descriptions (in particular it is the case for applications where graphs are naturally in normal form). But in some applications, each concept node has a specific situation in its SG^{ref} and a specific description; identifying these nodes or mixing their descriptions would lead to a loss of information. For instance, let c and c' be two concept nodes appearing in a NG^{ref} G and representing a given lake. c appears in the context of a biological study and its description contains biological information about the lake: animals and plants living in the lake. c' appears in the context of a touristic study and its description contains touristic information about the lake: possibilities of bathing, sailing or walking at the lake. The formulas $\Phi(G)$, $\Phi(H)$, and $\Phi(K)$ are equivalent, where H is obtained from G by exchanging the description graphs of c and c' and K is the k-normal form of G and H, with $k = max(depth(G), depth(H))$, in which the description of c and c' is the union of the biological and touristic descriptions. Such an equivalence is obviously undesirable. The semantics Ψ is better suited for these applications.

6 The Semantics Ψ

The semantics Ψ is extended from SG^{ref}s to NG^{ref}s in the same way as the semantics Φ, except that the context argument is the variable e_c assigned to the concept node c representing the context instead of the term $e_{\overline{c}}$. Thus a description is specific to a concept node rather than to a co-identity class. A context argument is added to each predicate (concept type predicates become 3-adic). For any NG^{ref} G and any node K of $A(G)$, let $nc(K)$ be the number of K concept nodes. Assign $r(G)$ variables y_1, ..., $y_{r(G)}$ to the $r(G)$ $co\text{-}ref_G$ classes and, for any node K of $A(G)$, assign $nc(K)$ variables x_1^K, ..., $x_{nc(K)}^K$ to K concept nodes. All variables y_i and x_j^K are distinct. The variables of $\Psi(K)$ are x_1^K, ..., $x_{nc(K)}^K$ and, if K contains generic nodes, some of the variables y_1, ..., $y_{r(G)}$. $\Psi'(t, G')$ is defined by induction on the depth of G'. For any node K of $A(G)$, $\Psi'(t, K)$ denotes the conjunction of the atoms obtained from those of $\Psi(K)$ by adding the first argument t; for any concept node c of K, e_c denotes the variable associated to c in $\Psi(K)$

$$\Psi'(t, G') = \exists x_1^{R(G')} ... x_{nc(R(G'))}^{R(G')}$$
$$(\Psi'(t, R(G')) \wedge (\wedge_{c \in D(G')} \Psi'(e_c, Desc(c))))$$

The formula

$$\Psi(\mathrm{G}) = \exists y_1 ... y_{r(G)} \Psi'(a_0, G)$$

is associated to any NG^{ref} G defined on the support S.

E.g. the formula associated to the graph G of Figure 3 is

$\Psi(G) = \exists y_2 ... y_5 y_7 y_8 (\exists x_1 ... x_5 (Person(a_0, Peter, x_1) \wedge Stick(a_0, y_2, x_2) \wedge Announcement(a_0, y_3, x_3) \wedge Billboard(a_0, y_4, x_4) \wedge Write(a_0, y_5, x_5) \wedge agent(a_0, x_2, x_1) \wedge object(a_0, x_2, x_3) \wedge location(a_0, x_2, x_4) \wedge agent(a_0, x_5, x_1) \wedge object(a_0, x_5, x_3) \wedge \Psi'(x_3, Desc(c_3))))$,

with

$\Psi'(x_3, Desc(c_3)) = \exists x_6 x_7 x_8 (Person(x_3, Peter, x_6) \wedge Sell(x_3, y_7, x_7) \wedge Bicycle(x_3, y_8, x_8) \wedge agent(x_3, x_7, x_6) \wedge object(x_3, x_7, x_8))$.

$\Psi(G)$ may be defined from $A(G)$ as the existential closure of the conjunction of the formulas $\Psi'(t_K, K)$, where K denotes a node of $A(G)$, $t_K = a_0$ if $K = R(G)$ and otherwise $t_K = e_c$, with $(J, c)K$ is the edge of $A(G)$ into K.

Projection is sound and complete with respect to Ψ.

Theorem 3 *Let G and H be two NG^{ref}s. G subsumes H iff $\Psi(S), \Psi(H) \models \Psi(G)$.*

Sketch of proof: A detailed proof is given in [Sim96a], similar to that of Theorem 2 concerning Φ. A more intuitive one is given here, which would not be available for Φ, as the SG^{ref} $Simple(H)$ defined below is not in normal form. Let S_0 be the support obtained from S by adding the individual marker a_0, a greatest concept type $\top$ (if it does not already exist) and a binary relation type $t_{context}$ (context relation). For any NG^{ref} G on S, let G_0 be the NG^{ref} on S_0 restricted to a concept node labelled $(\top, a_0, G)$ and let $Simple(G)$ be the SG^{ref} on S_0 obtained from the union of the nodes of $A(G_0)$ by adding for each edge $(J, c)K$ of $A(G_0)$ $nc(K)$ relation nodes of type $t_{context}$ which relate c to each concept node of K. It can be shown that (1) G subsumes H iff $Simple(G)$ subsumes $Simple(H)$ (as both propositions are equivalent to the existence of an A-projection from $A(G)$ to $A(H)$) and (2) $\Psi(S), \Psi(H) \models \Psi(G)$ iff $\Psi(S_0), \Psi(Simple(H)) \models \Psi(Simple(G))$ (as any s-substitution from G to H is a s-substitution from $Simple(G)$ to $Simple(H)$ and conversely). We conclude with the soundness and completeness result on SG^{ref}s. □

7 Related works

In the conceptual graph domain, several kinds of CGs with complex nodes have been used [CM97]. Most of them follow the formalism proposed by [Sow94]. Briefly, a specific relation type called ***description*** and a special concept type called *Graph* are introduced.

Concept nodes of type *Graph* may have a graph as referent. Adding an internal information D to a concept node c of a CG G is done by creating another concept node c' of type *Graph* and referent D, with c' and c related via a *description* relation. The mapping Φ for SGs is extended by introducing a special binary predicate *descr* — associated to the *description* relation — whose first argument is a term and second argument is a formula. $\Phi(G)$ contains the atom $descr(e,\ \Phi(D))$, where e is the term associated to c. This logical semantics is similar to the logical semantics of context of [McC93], [Guh91], with *descr* playing the same role as the *ist* predicate. [GW96] defines nested conceptual graphs where concept labels have three fields like ours. But these kinds of nested graphs include neither non-trivial graph-theoretic operations (such as projection) for reasoning purposes, nor FOL logical semantics. Another logical semantics for nested graphs, but without co-reference links, is given in [PMC95]. A specific logical language is proposed, where formulas do not belong to classical FOL, and a Gentzen system is defined, which is sound and complete for projection.

At first glance, there is some likeness between NGs and partitioned networks [Hen79]. But these two models actually only share a very general notion of grouping. NGs have been introduced for representing hierarchical knowledge and for doing reasonings respecting the levels. In partitioned networks, "the central idea of partioning is to allow groups of nodes and arcs to be bundled together into units called spaces, which are fundamental entities in partioned networks, on the same level as nodes and arcs. [..] Nodes and arcs of different spaces may be linked. [...] Every node and every arc of a network belongs to one or more space" [Hen79]. This quotation points out at least three important differences. First, in NGs, a nested graph is not on the same level as nodes or arcs. Second, nodes of different NGs cannot be individualy linked (unless by a co-reference link). Third, nested CGs cannot overlap. Thus, from a descriptive viewpoint, partioned networks are much more general than NGs. However, as far as we know, partioned networks are not provided with a morphism notion for doing reasonings.

There are possible links between the NG model and contextual reasoning, although contextual reasoning is not our goal here. Indeed, we developed the NG model for representing knowledge structured by hierarchical levels and doing reasonings which respect these levels. However, NGs may be used for representing a very particular case of contexts (the set of contexts is tree-structured, contextual formulas are restricted to existential positive conjunctive FOL formulas). But none of the logical semantics proposed in this paper seem to be close to one of the logical formalizations of contextual reasoning (e.g. [McC93], [Guh91], [Buv96]). However, more in-depth comparisons will now have to be done.

8 Conclusion

In this paper, a graph-based model for hierarchical structured knowledge, "nested graphs", is presented. Projection — a labelled nested graph morphism — is the main reasoning tool. The model is given two FOL semantics, which correspond to different intuitive semantics. Projection proved to be sound and complete with respect to deduction in both cases.

We are developing a workbench, called CoGITo, for building knowledge-based applications, in which every piece of knowledge is described by nested graphs (see [Hae95] [CCC+97] for a previous version based on simple graphs). This workbench forms the skeleton of a modelisation language for an industrial application we are currently involved in [BBV97].

Acknowledgements

We would like to specially thank two referees who drew our attention to the correspondence between the conjunctive query containment in databases and the simple graph projection, as well as M. C. Rousset, who suggested the transformation from NGs to SGs used in the proof of theorem 3.

References

[AHV95] S. Abiteboul, R. Hull, and V. Vianu. ***Foundations of Databases.*** Addison-Wesley, 1995.

[BBV97] C. Bos, B. Botella, and P. Vanheeghe. Modeling and simulating human behaviors with conceptual graphs. In *Conceptual Structures: Fulfilling Peirce's Dream, ICCS'97 Proc., LNAI 1257*, pages 275–289. Springer Verlag, 1997.

[Buv96] S. Buvač. Quantificational Logic of Context. In *Proc. AAAI'96*, pages 600—606, 1996.

[CCC+97] B. Carbonneill, M. Chein, O. Cogis, O. Guinaldo, O. Haemmerlé, M.L. Mugnier, and E. Salvat. The COnceptual Graphs at LIRMM Project. In *Proc. of the first CGTOOLS workshop*, pages 5–8, 1997.

[CM92] M. Chein and M.L. Mugnier. Conceptual Graphs: fundamental notions. *Revue d'Intelligence Artificielle*, 6(4):365–406, 1992.

[CM97] M. Chein and M.-L. Mugnier. Positive Nested Conceptual Graphs. In *Conceptual Structures: Fulfilling Peirce's Dream, ICCS'97 Proc., LNAI 1257*, pages 95–109. Springer Verlag, 1997.

[Dec92] R. Dechter. Constraint Networks. In *Encyclopedia of Artificial Intelligence*, pages 276—285. Wiley and Sons (second ed.), 1992.

[Guh91] R.V. Guha. Contexts: a formalization and some applications. PhD Thesis, Stanford University , 1991.

[GW96] B.C. Ghosh and V. Wuwongse. Computational Situation Theory in the Conceptual Graph Language. In *Lecture Notes in AI, 1115, Proc. ICCS'96*, pages 188–202, 1996.

[Hae95] O. Haemmerlé. *CoGITo : une plate-forme de développement de logiciels sur les graphes conceptuels*. PhD thesis, Université Montpellier II, 1995.

[Hen79] G.C. Hendrix. Encoding Knowledge in Partitioned Networks. In N. Findler, editor, *Associative Networks*, pages 51—92. Academic Press, 1979.

[LE91] R. Levinson and G. Ellis. Multi-Level Hierarchical Retrieval. In *Proc. 6th Annual Workshop on Conceptual Graphs*, pages 67–81, 1991.

[Leh92] F. Lehmann, editor. *Semantics Networks in Artificial Intelligence*. Pergamon Press, 1992.

[MC96] M.L. Mugnier and M. Chein. Représenter des connaissances et raisonner avec des graphes. *Revue d'Intelligence Artificielle*, 10(1):7–56, 1996.

[McC93] J. McCarthy. Notes on Formalizing Context. In *Proc. IJCAI'93*, pages 555–560, 1993.

[Mug95] M.L. Mugnier. On generalization/-specialization for conceptual graphs. *J. Expt. Theor. Artif. Intell.*, 7(3):325–344, 1995.

[PMC95] A. Preller, M.-L. Mugnier, and M. Chein. A logic for Nested Graphs. Research Report 95-038, LIRMM, 1995. To appear in Computational Intelligence.

[Rob92] D.D. Roberts. The Existential Graphs. In [Lehmann, 1992], pages 639–663. 1992.

[Sch91] L.K. Schubert. Semantic Networks are in the Eye of the Beholder. In J. F. Sowa, editor, *Principles of Semantic Networks*, pages 95—108. Morgan Kaufmann, 1991.

[SCM98] G. Simonet, M. Chein, and M.-L. Mugnier. Projection in Conceptual Graphs and Query Containment in nr-Datalog. Research Report 98-025, LIRMM, 1998.

[Sim96a] G. Simonet. Une autre sémantique logique pour les graphes emboités. Research Report 96-048, LIRMM, 1996.

[Sim96b] G. Simonet. Une sémantique logique pour les graphes emboités. Research Report 96-047, LIRMM, 1996.

[Sow76] J.F. Sowa. Conceptual graphs for a database interface. *IBM J. Research and development*, 20(4):336–357, 1976.

[Sow84] J.F. Sowa. *Conceptual Structures - Information Processing in Mind and Machine*. Addison-Wesley, 1984.

[Sow94] J.F. Sowa. Logical Foundations for Representing Object-oriented Systems. *J. Expt. Theor. Artif. Intell.*, 5, 1994.

[vdB94] H. van den Berg. Modal Logics for Conceptual Graphs III. In *Proc. ECAI*, pages 356—360, 1994.

Reasoning about Actions III

Situation Calculus and Causal Logic

Vladimir Lifschitz
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712, USA
vl@cs.utexas.edu

Abstract

In the AAAI-97 paper by McCain and Turner, the frame problem is solved using a nonmonotonic causal logic. In this note we show how their method can be adapted to the language of the situation calculus. The "causal situation calculus" is applied to action domains involving ramifications and nondeterminism, and related to the causal formalization of actions proposed by Lin.

1 Introduction

In this note, a nonmonotonic causal logic is used for describing actions in the situation calculus. This logic was defined for the propositional case in [McCain and Turner, 1997] on the basis of earlier work by several authors, including [Geffner, 1990], [Lin, 1995] and [McCain and Turner, 1995]. In [Lifschitz, 1997], it is extended to languages with variables. It is closely related to special cases of "general purpose" nonmonotonic formalisms such as logic programming, default logic and circumscription [Turner, 1998], and appears to be particularly well suited for the task of describing the effects of actions.

In [McCain and Turner, 1997], values of fluents are represented by propositional atoms indexed by time. For instance, the atom $Closed_5$ would express that the fluent *Closed* holds at time 5. Actions are treated in a similar way: the atom $Close_5$ would express that the action *Close* was performed between time instants 5 and 6. Action descriptions in [Lifschitz, 1997] follow the same pattern, except that when variables are available in the language, a fluent can be represented by a predicate symbol that takes time as the argument, or as one of several arguments. We can express, for instance, that the action of moving an object x to a location l was performed between times t and $t+1$ by the atomic formula $Move(x,l,t)$.

This "time-based" representation is different from the situation calculus approach [McCarthy and Hayes, 1969], which represents actions by terms and would use an expression like $Result(Move(x,l),s)$ to denote the situation that results from moving x to location l in situation s. Advantages and disadvantages of the ontology and syntax of the situation calculus have been the subject of many debates in the knowledge representation community. It is clear that at least in simple cases the language of the situation calculus is very attractive.

The purpose of this note is to show that the version of the nonmonotonic causal logic presented in [Lifschitz, 1997] is applicable in the framework of the situation calculus as well, that the elegant solution to the frame problem proposed in [McCain and Turner, 1997] can be easily expressed in the language of the situation calculus, and that the process of "literal completion" described in these papers may allow us to translate this solution into first-order logic.

To facilitate the comparison of the causal version of the situation calculus with earlier work, we apply it here to a formalization of the blocks world which is similar to the one presented in [Lifschitz, 1991]. This formalization involves actions with indirect effects. Then we discuss an enhancement of the basic example in which the effects of actions can be nondeterministic.

The description of the blocks world used in [Lifschitz, 1991] is outlined in Section 2. In Sections 3 and 4 we turn this description into a causal theory and show how this theory can be converted into a set of first-order axioms. The relationship between our method and Lin's is discussed in Section 5. The enhancement in which nondeterministic actions are included is presented in Section 6.

To make this paper self-contained, we reproduce the definition of the semantics of causal theories from [Lifschitz, 1997] in Appendix A. Appendix B contains proofs of theorems.

2 Indirect Effects of Moving a Block

We will describe the locations of blocks using the fluent expressions

$$On(x, y)$$

(block x is on block y) and

$$OnT(x)$$

(block x is on the table), which will be declared "inertial," that is, subject to the commonsense law of inertia. Fluent expressions of the form

$$Clear(x)$$

will be characterized by relating them to On, and they will not be inertial.

The effect of placing a block x on top of block y is described in [Lifschitz, 1991] by the axiom[1]

$$\begin{array}{l} Poss(Move(x, y), s) \\ \quad \supset Holds(On(x, y), Result(Move(x, y), s)). \end{array} \qquad (1)$$

On the other hand, a block cannot be in two places at once:

$$\begin{array}{l} Holds(On(x, y), s) \wedge Holds(On(x, z), s) \supset y = z, \\ \neg(Holds(On(x, y), s) \wedge Holds(OnT(x), s)). \end{array} \qquad (2)$$

In view of this fact, moving a block may have an "indirect" effect on an inertial fluent: performing the action $Move(x, y)$ will make $On(x, z)$ false if z is the current location of x and $y \neq z$, and it will make $OnT(x)$ false if x is currently on the table.

The presence of indirect effects makes the frame problem more difficult. (This aspect of the frame problem is known as the "ramification problem.") In [Lifschitz, 1991], this difficulty is addressed using circumscription in the style of [Baker, 1991]. The logical machinery that is employed there is rather complicated. It includes a second-order formulation of the existence of situation axioms and conjoining three circumscriptions with carefully selected lists of varied constants. In this note we show that the same goal can be achieved more easily if the effects of moving a block are described by causal rules.

3 A Causal Theory of the Blocks World

In causal logic, we distinguish between the claim that a proposition is true and the stronger claim that there is a *cause* for it to be true.

[1]The action of putting a block on the table is not part of this formalization, but it can be easily added.

Consider a language of classical logic, possibly many-sorted or higher-order. A *causal rule* is an expression of the form

$$F \leftarrow G \qquad (3)$$

where F and G are formulas of this language, called the *head* and the *body* of the rule. Intuitively, (3) expresses that F has a cause if G is true, or that G provides a "causal explanation" for F. If the body G of (3) is the logical constant $True$, we drop it and write the rule as $F \leftarrow$. A formula F is identified with the rule

$$False \leftarrow \neg F. \qquad (4)$$

A *causal theory* is defined by

- a finite set of nonlogical constants[2] of the underlying language, called the *explainable symbols* of the theory, and
- a finite set of causal rules.

The semantics of causal theories is defined in [Lifschitz, 1997] by a translation into classical logic; a causal theory is viewed as shorthand for the sentence obtained as the result of this translation. In particular, a *model* of a causal theory T is a model of its translation, in the sense of classical logic; a *theorem* of T is a sentence that is logically entailed by the translation of T. The definition of the translation is reproduced in Appendix A. Its main feature is that it can be written as the conjunction of the universal closures of the implications

$$G \supset F \qquad (5)$$

for all rules (3) of the theory plus an additional conjunctive term. (This term makes the translation nonmonotonic, just like the minimality condition in the definition of circumscription.) In this sense, a causal rule (3) is stronger than the corresponding implication (5).

We will define now a causal theory of the blocks world, CBW. The explainable symbols of the theory are the predicate constants $Holds$ and $Inertial$.

Formula (1) is replaced in CBW by the rule

$$\begin{array}{l} Holds(On(x, y), Result(Move(x, y), s)) \\ \qquad \leftarrow Poss(Move(x, y), s). \end{array} \qquad (6)$$

Thus instead of postulating merely that the fluent $On(x, y)$ holds after performing $Move(x, y)$, we say that there is a *cause* for this fluent to hold.

[2]A *nonlogical constant* is a function constant or a predicate constant (other than equality).

Formula (2) is replaced by the rules saying that, wherever a block is located, there is a cause for it not to be anywhere else:

$$\begin{aligned}\neg Holds(On(x,z),s) &\leftarrow Holds(On(x,y),s) \land y \neq z,\\ \neg Holds(OnT(x),s) &\leftarrow Holds(On(x,y),s),\\ \neg Holds(On(x,y),s) &\leftarrow Holds(OnT(x),s).\end{aligned} \tag{7}$$

The predicate constant *Inertial* is characterized by the rules

$$\begin{aligned}Inertial(On(x,y)) &\leftarrow ,\\ Inertial(OnT(x)) &\leftarrow ,\\ \neg Inertial(f) &\leftarrow \neg Inertial(f).\end{aligned} \tag{8}$$

The last rule says that if a fluent is not inertial then there is a cause for this. Proposition 1 from [Lifschitz, 1997] shows that including this rule has the same effect as circumscribing *Inertial*.

In accordance with a proposal from [McCain and Turner, 1997], the commonsense law of inertia is understood as the assertion that if an inertial fluent does not change its value after an action is executed then there is a cause for this:

$$\begin{aligned}Holds(f,Result(a,s)) \leftarrow\ & Inertial(f) \land Poss(a,s)\\ & \land Holds(f,s)\\ & \land Holds(f,Result(a,s)),\\ \neg Holds(f,Result(a,s)) \leftarrow\ & Inertial(f) \land Poss(a,s)\\ & \land \neg Holds(f,s)\\ & \land \neg Holds(f,Result(a,s)).\end{aligned} \tag{9}$$

Rules (6), (7) and (9) provide causal explanations for the truth values of $Holds(f,s)$ only when f is inertial and s is the result of performing an action. In all other cases, we postulate that, whatever the truth value of $Holds(f,s)$ is, there is a cause for this:

$$\begin{aligned}Holds(f,s) &\leftarrow Holds(f,s) \land \neg Inertial(f),\\ \neg Holds(f,s) &\leftarrow \neg Holds(f,s) \land \neg Inertial(f),\\ Holds(f,s) &\leftarrow Holds(f,s) \land \neg\exists a s_1(Poss(a,s_1)\\ &\qquad \land s = Result(a,s_1)),\\ \neg Holds(f,s) &\leftarrow \neg Holds(f,s) \land \neg\exists a s_1(Poss(a,s_1)\\ &\qquad \land s = Result(a,s_1)).\end{aligned} \tag{10}$$

In addition to these causal rules, we need a few postulates that are expressed by formulas. (Recall that a formula can be viewed as a causal rule (4) whose head is the logical constant *False*.) One of these formulas tells us when the action $Move(x,y)$ is executable:

$$\begin{aligned}Poss(Move(x,y),s) \equiv\ & Holds(Clear(x),s)\\ & \land Holds(Clear(y),s)\\ & \land x \neq y.\end{aligned} \tag{11}$$

We include also the definition of *Clear* in terms of *On*:

$$Holds(Clear(x),s) \equiv \neg\exists y Holds(On(y,x),s). \tag{12}$$

The remaining postulates express the unique name assumption. The terms $On(x,y)$, $OnT(x)$ and $Clear(x)$ represent distinct fluents:

$$\begin{aligned}On(x_1,y_1) = On(x_2,y_2) &\supset x_1 = x_2 \land y_1 = y_2,\\ OnT(x) = OnT(y) &\supset x = y,\\ Clear(x) = Clear(y) &\supset x = y,\\ On(x,y) &\neq OnT(z),\\ On(x,y) &\neq Clear(z),\\ OnT(x) &\neq Clear(y).\end{aligned} \tag{13}$$

The terms $Move(x,y)$ represent distinct actions:

$$Move(x_1,y_1) = Move(x_2,y_2) \supset x_1 = x_2 \land y_1 = y_2. \tag{14}$$

A situation contains complete information about its past:

$$Result(a_1,s_1) = Result(a_2,s_2) \supset a_1 = a_2 \land s_1 = s_2. \tag{15}$$

This is the end of the list of rules of *CBW*.

4 Simplifying *CBW*

According to the semantics of causal theories, *CBW* is shorthand for a sentence of classical logic that includes second-order quantifiers. The process of literal completion allows us to rewrite this sentence in first-order logic:

Proposition 1. *CBW is equivalent to the conjunction of the universal closures of formulas (1), (2), (11)–(15),*

$$\begin{aligned}&Inertial(f)\\ &\quad \equiv \exists xy(f = On(x,y)) \lor \exists x(f = OnT(x)),\end{aligned} \tag{16}$$

$$\begin{aligned}&Inertial(f) \land Poss(a,s)\\ &\quad \land \neg Holds(f,s) \land Holds(f,Result(a,s))\\ &\qquad \supset \exists xy[f = On(x,y) \land a = Move(x,y)]\end{aligned} \tag{17}$$

and

$$\begin{aligned}&Inertial(f) \land Poss(a,s)\\ &\land Holds(f,s) \land \neg Holds(f,Result(a,s))\\ &\quad \supset \exists xyz[f = On(x,z) \land Holds(On(x,y),Result(a,s))\\ &\qquad \land y \neq z]\\ &\quad \lor \exists xy[f = OnT(x) \land Holds(On(x,y),Result(a,s))]\\ &\quad \lor \exists xy[f = On(x,y) \land Holds(OnT(x),Result(a,s))].\end{aligned} \tag{18}$$

Formula (16) is an explicit definition of *Inertial*. Formula (17) is an explanation closure axiom in the sense of [Schubert, 1990] and [Reiter, 1991]: it tells us under what condition a fluent f can change its value from *false* to *true* after executing an action. It says that, for an inertial fluent, this can only happen if the fluent has the form $On(x,y)$, and if the action that has been executed is $Move(x,y)$.

Formula (18) can be viewed as a second explanation closure axiom—it tells us under what condition f can change its value from *true* to *false*. But it does this in a somewhat indirect way, without actually saying which action has been executed. We can be more explicit: an inertial fluent can change its value from *true* to *false* after executing an action only if

- the fluent has the form $On(x,y)$, and the action consisted in placing x on top of a block other than y, or
- the fluent has the form $OnT(x)$, and the action consisted in placing x on top of a block.

The following corollary to Proposition 1 shows that this is indeed an equivalent reformulation of (18):

Corollary. *CBW is equivalent to the conjunction of the universal closures of formulas (1), (2), (11)–(17) and*

$$\begin{array}{l} Inertial(f) \wedge Poss(a,s) \\ \quad \wedge Holds(f,s) \wedge \neg Holds(f, Result(a,s)) \\ \qquad \supset \exists xyz[f = On(x,y) \wedge a = Move(x,z) \wedge y \neq z] \\ \qquad\quad \vee \exists xy[f = OnT(x) \wedge a = Move(x,y)]. \end{array} \tag{19}$$

5 Comparison with Lin's Method

Lin [1995] expresses the basic distinction between being true and having a cause without introducing a specialized nonmonotonic formalism. Instead, he extends the situation calculus language by the new predicate *Caused*, and applies circumscription to minimize this predicate. In this section, we define a Lin-style formalization of the blocks world and relate it to the causal theory *CBW*.

The predicate *Caused* has three arguments: a fluent f, a truth value v, and a situation s in which there is a cause for f to have the value v. It is convenient to replace *Caused* by two binary predicates corresponding to the two possible values of v: $Caused^+(f,s)$ will express that there is a cause for f to be true in the situation s, and $Caused^-(f,s)$ will express that there is a cause for f to be false.

The formulas corresponding to rules (6) and (7) in this formalism are

$$\begin{array}{l} Poss(Move(x,y),s) \\ \quad \supset Caused^+(On(x,y), Result(Move(x,y),s)) \end{array} \tag{20}$$

and

$$\begin{array}{l} Holds(On(x,y),s) \wedge y \neq z \supset Caused^-(On(x,z),s), \\ \quad Holds(On(x,y),s) \supset Caused^-(OnT(x),s), \\ \quad Holds(OnT(x),s) \supset Caused^-(On(x,y),s). \end{array} \tag{21}$$

The predicate *Inertial* is described by the axioms

$$\begin{array}{l} Inertial(On(x,y)), \\ Inertial(OnT(x)). \end{array} \tag{22}$$

By *LBW* we denote the parallel circumscription of $Caused^+$, $Caused^-$ and *Inertial* in the conjunction of the universal closures of (20)–(22).

To make the result of this circumscription useful for determining the values of fluents, we need to relate the new predicates $Caused^+$ and $Caused^-$ to *Holds* by the axioms

$$\begin{array}{l} Caused^+(f,s) \supset Holds(f,s), \\ Caused^-(f,s) \supset \neg Holds(f,s), \end{array} \tag{23}$$

and to add an axiom expressing the commonsense law of inertia:

$$\begin{array}{l} Inertial(f) \wedge Poss(a,s) \\ \quad \wedge \neg Caused^+(f, Result(a,s)) \\ \quad \wedge \neg Caused^-(f, Result(a,s)) \\ \qquad \supset [Holds(f, Result(a,s)) \equiv Holds(f,s)]. \end{array} \tag{24}$$

Alternatively, inertia can be stated in the explanation closure format:

$$\begin{array}{l} Inertial(f) \wedge Poss(a,s) \\ \quad \wedge \neg Holds(f,s) \wedge Holds(f, Result(a,s)) \\ \qquad \supset Caused^+(f, Result(a,s)), \\ Inertial(f) \wedge Poss(a,s) \\ \quad \wedge Holds(f,s) \wedge \neg Holds(f, Result(a,s)) \\ \qquad \supset Caused^-(f, Result(a,s)). \end{array} \tag{25}$$

It is easy to see that the equivalence between (24) and the conjunction of the formulas (25) is a propositional consequence of (23).

The relationship between this formalization of the blocks world and the causal theory *CBW* presented in Section 3 is described in the following proposition:

Proposition 2. *The conjunction of LBW with the universal closures of (11)–(15), (23) and (25) is equivalent to the conjunction of CBW with the universal closures of*

$$\begin{array}{l} Caused^+(f,s) \\ \quad \equiv \exists xys'[Poss(Move(x,y),s') \wedge f = On(x,y) \\ \qquad \wedge s = Result(Move(x,y),s')] \end{array} \tag{26}$$

and

$$\begin{aligned}&Caused^-(f,s)\\ &\quad \equiv \exists xyz[f = On(x,z) \wedge Holds(On(x,y),s)\\ &\qquad\qquad \wedge y \neq z] \qquad (27)\\ &\qquad \vee \exists xy[f = OnT(x) \wedge Holds(On(x,y),s)]\\ &\qquad \vee \exists xy[f = On(x,y) \wedge Holds(OnT(x),s)].\end{aligned}$$

Thus the Lin-style causal formalization of the blocks world is equivalent to the definitional (and consequently conservative) extension of *CBW* obtained by adding the definitions (26), (27) of $Caused^+$ and $Caused^-$.

The two formalizations are also similar in that transforming each of them into an equivalent first-order theory is accomplished by a completion process.

On the other hand, these formalizations differ from each other in their approach to inertia. Formula (24), similar to the formalization of the commonsense law of inertia in [Lin, 1995], treats it monotonically: this formula is included as an additional constraint on top of a circumscriptive formalization of causality. By dropping (24) from the list of axioms we would make the theory weaker. Rules (9), which express inertia in *CBW* and are inspired by [McCain and Turner, 1997], operate nonmonotonically. By dropping them from the list of rules we would make the theory stronger—it would become inconsistent.

6 Nondeterministic Actions

Imagine that a block can be damaged when it is moved to a new location. The effect of the action $Move(x,y)$ on the new fluent $Damaged(x)$ is nondeterministic: if this fluent is false in a situation s then in the new situation $Result(Move(x,y),s)$ it may become true, but it may also remain false.

To express that a block is *always* damaged when it is moved to a new location, we would write the rule

$$\begin{aligned}&Holds(Damaged(x), Result(Move(x,y),s))\\ &\qquad\qquad \leftarrow Poss(Move(x,y),s)),\end{aligned}$$

similar to rule (6). In accordance with the approach to nondeterministic actions proposed in [McCain and Turner, 1997], to express that a block *may be* damaged after performing this action, we append a copy of the head of this rule to the body as another conjunctive term:

$$\begin{aligned}&Holds(Damaged(x), Result(Move(x,y),s))\\ &\quad \leftarrow Poss(Move(x,y),s)\\ &\qquad \wedge Holds(Damaged(x), Result(Move(x,y),s)). \qquad (28)\end{aligned}$$

Rule (28) says that if x damaged after performing the action then there is a cause for this. The head of the rule is asserted here to be a causally justified possibility.

The enhancement CBW' of the theory *CBW* from Section 3 in which we can talk about damaged blocks is obtained by adding rule (28), the rule

$$Inertial(Damaged(x)) \leftarrow \qquad (29)$$

and the unique name axioms

$$\begin{aligned}Damaged(x) = Damaged(y) &\supset x = y,\\ Damaged(x) &\neq On(y,z),\\ Damaged(x) &\neq OnT(y),\\ Damaged(x) &\neq Clear(y).\end{aligned} \qquad (30)$$

Proposition 3. *CBW' is equivalent to the conjunction of the universal closures of formulas (1), (2), (11)–(15), (18), (30),*

$$\begin{aligned}&Inertial(f)\\ &\quad \equiv \exists xy(f = On(x,y)) \vee \exists x(f = OnT(x))\\ &\qquad\qquad \vee \exists x(f = Damaged(x))\end{aligned} \qquad (31)$$

and

$$\begin{aligned}&Inertial(f) \wedge Poss(a,s)\\ &\quad \wedge \neg Holds(f,s) \wedge Holds(f, Result(a,s))\\ &\qquad \supset \exists xy[f = On(x,y) \wedge a = Move(x,y)]\\ &\qquad\quad \vee \exists xy[f = Damaged(x) \wedge a = Move(x,y)].\end{aligned} \qquad (32)$$

The last formula differs from the explanation closure condition (17) by the presence of the disjunctive term that describes how a block can be damaged. This is the main difference between the characterization of *CBW* in Proposition 1 and the characterization of CBW' in Proposition 3.

7 Future Work

The examples presented above show how the causal situation calculus handles ramifications and nondeterminism. Otherwise, the expressive possibilities of this formalism remain unexplored. One particularly interesting issue is the treatment of qualification constraints in the sense of [Lin and Reiter, 1994].

Methodologies for representing actions can be compared in a precise and general way by proving embedding theorems, such as the results presented in [Kartha, 1993]. Our plans for future work include relating the causal situation calculus to the action language $\mathcal{C}$ from [Giunchiglia and Lifschitz, 1998].

Acknowledgements

I am grateful to Enrico Giunchiglia, Fangzhen Lin, Norman McCain and Hudson Turner for useful comments. This work was partially supported by National Science Foundation under grant IRI-9306751.

Appendix A: Semantics of Causal Theories

In the definition of the semantics of causal theories below, we use the substitution of variables for the explainable symbols in a formula. In connection with this, it is convenient to denote formulas by expressions like $F(E)$, where E is the list of all explainable symbols. Then, for any tuple e of variables that is similar[3] to E, the result of replacing all occurrences of the constants E in $F(E)$ by the variables e can be denoted by $F(e)$.

Consider a causal theory T with the explainable symbols E and the causal rules

$$F_i(E, x^i) \leftarrow G_i(E, x^i) \qquad (i = 1, \ldots),$$

where x^i is the list of all free variables of the i-th rule. Take a tuple e of new variables similar to E. By $T^\bullet(e)$ we denote the formula

$$\bigwedge_i \forall x^i (G_i(E, x^i) \supset F_i(e, x^i)).$$

Note that the occurrences of explainable symbols in the heads are replaced here by variables, and the occurrences in the bodies are not. According to the semantics of causal theories, T is shorthand for the sentence

$$\forall e (T^\bullet(e) \equiv e = E). \qquad (33)$$

(The expression $e = E$ stands for the conjunction of the equalities between the members of e and the corresponding members of E.)

Intuitively, the condition $T^\bullet(e)$ expresses that the possible values e of the explainable symbols E are "explained" by the rules of T. Sentence (33) says that the actual values of these symbols are the only ones that are explained by the rules of T.

Formula (33) can be equivalently written as

$$T^\bullet(E) \wedge \forall e (T^\bullet(e) \supset e = E). \qquad (34)$$

In this expression, the term $T^\bullet(E)$ is the conjunction of the universal closures of the implications

$$G_i(E, x^i) \supset F_i(E, x^i)$$

[3]The similarity condition means that (i) e has the same length as E, (ii) if the k-th member of E is a function constant then the k-th member of e is a function variable of the same type, and (iii) if the k-th member of E is a predicate constant then the k-th member of e is a predicate variable of the same type.

corresponding to the rules of T. The second conjunctive term of (34) is what makes a causal rule stronger than the corresponding implication, and what makes the formalism nonmonotonic.

Appendix B: Proofs of Theorems

B.1 Literal Completion

As defined in [Lifschitz, 1997], a causal theory T is *definite* if every explainable symbol of T is a predicate constant, and the head of every rule of T

- is a literal that does not contain explainable symbols in argument positions, or
- contains no explainable symbols.

It is clear that CBW is definite.

The process of "literal completion" described in [McCain and Turner, 1997] for the propositional case and generalized in [Lifschitz, 1997], turns any definite causal theory into an equivalent set of sentences of classical logic, called its *completion sentences*. If the heads and the bodies of all rules of the theory are first-order formulas, as in CBW, then the completion sentences are first-order also. Applying literal completion is all that needs to be done to eliminate the second-order quantifiers from the sentence representing CBW.

Literal completion is similar to the process of completion familiar from logic programming ([Clark, 1978], [Lloyd and Topor, 1984]). Its main new feature is that the "completed definition" of an explainable predicate constant P consists of *two* equivalences, one "positive" and one "negative." Positive completion sentences are obtained from the rules whose heads are positive literals in exactly the same way as in [Lloyd and Topor, 1984]. Negative completion sentences are generated in a similar way from the rules whose heads are negative literals. In addition, for every rule $F \leftarrow G$ whose head F does not contain explainable symbols, there is a corresponding completion sentence, which is simply the universal closure of $G \supset F$.

In application to CBW, the literal completion procedure generates

(i) the positive completion sentence for *Holds*,

(ii) the negative completion sentence for *Holds*,

(iii) the positive completion sentence for *Inertial*,

(iv) the negative completion sentence for *Inertial*,

(v) the completion sentences corresponding to formulas (11)–(15).

We will consider all these sentences beginning with the easiest, those in group (v). The completion sentence for a rule of form (4) is the universal closure of

$$\neg F \supset \mathit{False},$$

which is equivalent to the universal closure of F. Consequently, sentences (v) are equivalent to the universal closures of (11)-(15).

Sentence (iii) is generated by completing the first two of rules (8). It is equivalent to the universal closure of (16). Sentence (iv) is generated in the same way from the third of rules (8):

$$\forall f(\neg \mathit{Inertial}(f) \equiv \neg \mathit{Inertial}(f)).$$

It is logically valid, and thus can be dropped from the conjunction of (i)-(v).

Sentence (i) is generated from rule (6), the first of rules (9), and the first and third of rules (10). It is equivalent to the universal closure of

$$\begin{aligned}
&\mathit{Holds}(f,s) \equiv \\
&\quad \exists xys_1[f = \mathit{On}(x,y) \wedge s = \mathit{Result}(\mathit{Move}(x,y),s_1) \\
&\qquad \wedge \mathit{Poss}(\mathit{Move}(x,y),s_1)] \\
&\quad \vee \exists as_1[s = \mathit{Result}(a,s_1) \wedge \mathit{Inertial}(f) \\
&\qquad \wedge \mathit{Poss}(a,s_1) \wedge \mathit{Holds}(f,s_1) \wedge \mathit{Holds}(f,s)] \\
&\quad \vee[\mathit{Holds}(f,s) \wedge \neg \mathit{Inertial}(f)] \\
&\quad \vee[\mathit{Holds}(f,s) \wedge \neg \exists as_1(\mathit{Poss}(a,s_1) \\
&\qquad \wedge s = \mathit{Result}(a,s_1))].
\end{aligned} \tag{35}$$

Sentence (ii) is generated from rules (7), the second of rules (9), and the second and fourth of rules (10). It is equivalent to the universal closure of

$$\begin{aligned}
&\neg \mathit{Holds}(f,s) \equiv \\
&\quad \exists xyz[f = \mathit{On}(x,z) \wedge \mathit{Holds}(\mathit{On}(x,y),s) \wedge y \neq z] \\
&\quad \vee \exists xy[f = \mathit{OnT}(x) \wedge \mathit{Holds}(\mathit{On}(x,y),s)] \\
&\quad \vee \exists xy[f = \mathit{On}(x,y) \wedge \mathit{Holds}(\mathit{OnT}(x),s)] \\
&\quad \vee \exists as_1[s = \mathit{Result}(a,s_1) \wedge \mathit{Inertial}(f) \\
&\qquad \wedge \mathit{Poss}(a,s_1) \wedge \neg \mathit{Holds}(f,s_1) \wedge \neg \mathit{Holds}(f,s)] \\
&\quad \vee[\neg \mathit{Holds}(f,s) \wedge \neg \mathit{Inertial}(f)] \\
&\quad \vee[\neg \mathit{Holds}(f,s) \wedge \neg \exists as_1(\mathit{Poss}(a,s_1) \\
&\qquad \wedge s = \mathit{Result}(a,s_1))].
\end{aligned} \tag{36}$$

Thus CBW is equivalent to the conjunction of the universal closures of formulas (11)-(16), (35) and (36).

B.2 Proof of Proposition 1

Proposition 1. *CBW is equivalent to the conjunction of the universal closures of formulas (1), (2) and (11)-(19).*

Each of the equivalences (35) and (36) can be written as the conjunction of two implications—left-to-right and right-to-left. We will denote these implications by $(35)^{\supset}$, $(35)^{\subset}$, $(36)^{\supset}$ and $(36)^{\subset}$.

By $\tilde{\forall} F$ we denote the universal closure of a formula F. We say that formulas F and G are *universally equivalent* if $\tilde{\forall} F$ is logically equivalent to $\tilde{\forall} G$. We say that formulas F and G are *universally equivalent in the presence of* a formula H if $\tilde{\forall} F \wedge \tilde{\forall} H$ is logically equivalent to $\tilde{\forall} G \wedge \tilde{\forall} H$, and similarly for several formulas $H_1, H_2, \ldots$.

In view of the assertion stated at the end of Section B.1, Proposition 1 will be proved if we establish the following lemmas:

Lemma 1. *$(35)^{\subset}$ is universally equivalent to (1).*

Lemma 2. *$(36)^{\subset}$ is universally equivalent to (2).*

Lemma 3. *In the presence of (15), formula $(35)^{\supset}$ is universally equivalent to (17).*

Lemma 4. *In the presence of (15), formula $(36)^{\supset}$ is universally equivalent to (18).*

Proof of Lemma 1. The antecedent of $(35)^{\subset}$ is the disjunction of four formulas, so that $(35)^{\subset}$ can be written as the conjunction of four implications. The first of them is

$$\begin{aligned}
\exists xys_1[f = \mathit{On}(x,y) \wedge s = \mathit{Result}(\mathit{Move}(x,y),s_1) \\
\wedge \mathit{Poss}(\mathit{Move}(x,y),s_1)] \\
\supset \mathit{Holds}(f,s),
\end{aligned}$$

which is universally equivalent to (1). The other three implications are tautological.

The proof of Lemma 2 is similar.

Proof of Lemma 3. $(35)^{\supset}$ is equivalent to

$$\begin{aligned}
&\mathit{Holds}(f,s) \\
&\supset \exists xys_1[f = \mathit{On}(x,y) \wedge s = \mathit{Result}(\mathit{Move}(x,y),s_1) \\
&\qquad \wedge \mathit{Poss}(\mathit{Move}(x,y),s_1)] \\
&\vee \exists as_1[s = \mathit{Result}(a,s_1) \wedge \mathit{Inertial}(f) \\
&\qquad \wedge \mathit{Poss}(a,s_1) \wedge \mathit{Holds}(f,s_1)] \\
&\vee \neg \mathit{Inertial}(f) \\
&\vee \neg \exists as_1(\mathit{Poss}(a,s_1) \wedge s = \mathit{Result}(a,s_1))
\end{aligned}$$

which is universally equivalent to

$$\begin{aligned}
&\mathit{Holds}(f,s) \wedge \mathit{Inertial}(f) \\
&\quad \wedge \mathit{Poss}(a,s_1) \wedge s = \mathit{Result}(a,s_1) \\
&\qquad \supset \exists xys_1[f = \mathit{On}(x,y) \\
&\qquad\qquad \wedge s = \mathit{Result}(\mathit{Move}(x,y),s_1) \\
&\qquad\qquad \wedge \mathit{Poss}(\mathit{Move}(x,y),s_1)] \\
&\qquad \vee \exists as_1[s = \mathit{Result}(a,s_1) \wedge \mathit{Poss}(a,s_1) \\
&\qquad\qquad \wedge \mathit{Holds}(f,s_1)].
\end{aligned}$$

In the presence of (15), this is universally equivalent to

$$\begin{aligned} & Holds(f,s) \land Inertial(f) \\ & \quad \land Poss(a,s_1) \land s = Result(a,s_1) \\ & \qquad \supset \exists xy[f = On(x,y) \land a = Move(x,y) \\ & \qquad\qquad \land Poss(Move(x,y),s_1)] \\ & \qquad \lor [Poss(a,s_1) \land Holds(f,s_1)] \end{aligned}$$

and consequently to

$$\begin{aligned} & Holds(f,s) \land Inertial(f) \\ & \quad \land Poss(a,s_1) \land s = Result(a,s_1) \\ & \qquad \supset \exists xy[f = On(x,y) \land a = Move(x,y)] \\ & \qquad\quad \lor Holds(f,s_1). \end{aligned}$$

This formula is universally equivalent to

$$\begin{aligned} & Holds(f,Result(a,s_1)) \land Inertial(f) \land Poss(a,s_1) \\ & \supset \exists xy[f = On(x,y) \land a = Move(x,y)] \lor Holds(f,s_1) \end{aligned}$$

and consequently to (17).

Proof of Lemma 4. $(36)^{\supset}$ is equivalent to

$$\begin{aligned} & \neg Holds(f,s) \\ & \supset \exists xyz[f = On(x,z) \land Holds(On(x,y),s) \land y \neq z] \\ & \quad \lor \exists xy[f = OnT(x) \land Holds(On(x,y),s)] \\ & \quad \lor \exists xy[f = On(x,y) \land Holds(OnT(x),s)] \\ & \quad \lor \exists as_1[s = Result(a,s_1) \land Inertial(f) \\ & \qquad \land Poss(a,s_1) \land \neg Holds(f,s_1)] \\ & \quad \lor \neg Inertial(f) \\ & \lor \neg \exists as_1(Poss(a,s_1) \land s = Result(a,s_1)) \end{aligned}$$

which is universally equivalent to

$$\begin{aligned} & \neg Holds(f,s) \land Inertial(f) \land Poss(a,s_1) \\ & \land s = Result(a,s_1) \\ & \quad \supset \exists xyz[f = On(x,z) \land Holds(On(x,y),s) \land y \neq z] \\ & \quad \lor \exists xy[f = OnT(x) \land Holds(On(x,y),s)] \\ & \quad \lor \exists xy[f = On(x,y) \land Holds(OnT(x),s)] \\ & \quad \lor \exists as_1[s = Result(a,s_1) \land Poss(a,s_1) \\ & \qquad\qquad \land \neg Holds(f,s_1)]. \end{aligned}$$

In the presence of (15), this is universally equivalent to

$$\begin{aligned} & \neg Holds(f,s) \land Inertial(f) \land Poss(a,s_1) \\ & \land s = Result(a,s_1) \\ & \quad \supset \exists xyz[f = On(x,z) \land Holds(On(x,y),s) \land y \neq z] \\ & \quad \lor \exists xy[f = OnT(x) \land Holds(On(x,y),s)] \\ & \quad \lor \exists xy[f = On(x,y) \land Holds(OnT(x),s)] \\ & \quad \lor [Poss(a,s_1) \land \neg Holds(f,s_1)] \end{aligned}$$

and consequently to

$$\begin{aligned} & Holds(f,s_1) \land \neg Holds(f,s) \\ & \land Inertial(f) \land Poss(a,s_1) \land s = Result(a,s_1)) \\ & \quad \supset \exists xyz[f = On(x,z) \land Holds(On(x,y),s) \land y \neq z] \\ & \quad \lor \exists xy[f = OnT(x) \land Holds(On(x,y),s)] \\ & \quad \lor \exists xy[f = On(x,y) \land Holds(OnT(x),s)]. \end{aligned}$$

This is universally equivalent to (18).

B.3 Proof of Corollary to Proposition 1

Corollary. *CBW is equivalent to the conjunction of the universal closures of formulas (1), (2), (11)–(17) and (19).*

It is sufficient to establish the following lemma:

Lemma 5. *The universal closures of (1), (2) and (11)–(17) entail the formulas*

$$\begin{aligned} & Poss(a,s) \land Holds(On(x,z),s) \land y \neq z \\ & \supset [Holds(On(x,y),Result(a,s)) \equiv a = Move(x,y))], \end{aligned} \tag{37}$$

$$\begin{aligned} & Poss(a,s) \land Holds(OnT(x),s) \\ & \supset [Holds(On(x,y),Result(a,s)) \equiv a = Move(x,y))] \end{aligned} \tag{38}$$

and

$$\begin{aligned} & Poss(a,s) \land Holds(On(x,y),s) \\ & \qquad \supset \neg Holds(OnT(x),Result(a,s)). \end{aligned} \tag{39}$$

Indeed, the only difference between the statements of Propositions 1 and of the corollary is that (18) is replaced in the latter by (19). By Lemma 5, (18) can be equivalently rewritten as

$$\begin{aligned} & Inertial(f) \land Poss(a,s) \\ & \land Holds(f,s) \land \neg Holds(f,Result(a,s)) \\ & \quad \supset \exists xyz[f = On(x,z) \land a = Move(x,y) \land y \neq z] \\ & \quad \lor \exists xy[f = OnT(x) \land a = Move(x,y)] \\ & \quad \lor False \end{aligned}$$

which is equivalent to (19).

Proof of Lemma 5. Formula (37) can be written as the conjunction of two, with the equivalence replaced by the implication left-to-right and by the implication right-to-left. We will denote these formulas by $(37)^{\supset}$ and $(37)^{\subset}$. Similarly, (38) can be divided into two parts $(38)^{\supset}$ and $(38)^{\subset}$.

Both $(37)^{\subset}$ and $(38)^{\subset}$ are entailed by (1). From (16) and (17),

$$\begin{aligned} & Poss(a,s) \land \neg Holds(On(x,y),s) \\ & \land Holds(On(x,y),Result(a,s)) \\ & \quad \supset \exists x_1y_1[On(x,y) = On(x_1,y_1) \land a = Move(x_1,y_1)] \end{aligned}$$

and

$$\begin{aligned} & Poss(a,s) \land \neg Holds(OnT(x),s) \\ & \land Holds(OnT(x),Result(a,s)) \\ & \quad \supset \exists x_1y_1[OnT(x) = On(x_1,y_1) \land a = Move(x_1,y_1)]. \end{aligned}$$

Then, by (13),

$$\begin{aligned} & Poss(a,s) \land \neg Holds(On(x,y),s) \\ & \quad \land Holds(On(x,y),Result(a,s)) \\ & \qquad \supset a = Move(x,y)) \end{aligned} \tag{40}$$

and

$$\neg[Poss(a,s) \wedge \neg Holds(OnT(x),s) \\ \wedge Holds(OnT(x), Result(a,s))]. \qquad (41)$$

Both $(37)^{\supset}$ and $(38)^{\supset}$ are immediate from (2) and (40); (39) is equivalent to (41).

B.4 Proof of Proposition 2

Proposition 2. *The conjunction of LBW with the universal closures of (11)–(15), (23) and (25) is equivalent to the conjunction of CBW with the universal closures of (26) and (27).*

The circumscription *LBW* is characterized by the following lemma, proved below:

Lemma 6. *LBW is equivalent to the conjunction of the universal closures of (16), (26) and (27).*

On the other hand, by Proposition 1, *CBW* is equivalent to the conjunction of the universal closures of formulas (1), (2) and (11)–(18). In view of these two facts, we only need to show that, in the presence of the "common axioms" (11)–(16), (26) and (27), the conjunction of (23) and (25) is universally equivalent to the conjunction of (1), (2), (17) and (18). It is easy to check that in the presence of the common axioms

- the first of formulas (23) is universally equivalent to (1),
- the second of formulas (23) is universally equivalent to (2),
- the first of formulas (25) is universally equivalent to (17), and
- the second of formulas (25) is universally equivalent to (18).

Proof of Lemma 6. Recall that *LBW* is the parallel circumscription of *Caused*$^+$, *Caused*$^-$ and *Inertial* in the conjunction of the universal closures of formulas (20)–(22). All occurrences of these predicate constants in (20)–(22) are positive. According to Proposition 7.1.1 from [Lifschitz, 1994], it follows that *LBW* is equivalent to the conjunction of the circumscriptions of each of the three constants in the same sentence. Each of these circumscriptions can be computed using the completion method (see [Lifschitz, 1994], Section 3.1), and the result of the computation is represented by formulas (16), (26) and (27).

B.5 Proof of Proposition 3

Proposition 3. *CBW′ is equivalent to the conjunction of the universal closures of formulas (1), (2), (11)–(15), (18), and (30)–(32).*

As discussed in Section B.1, literal completion turns *CBW* into the equivalent set of sentences (i)–(v). Let us consider how this result is affected by the presence of the additional postulates (28)–(30).

Group (v) includes now the universal closures of formulas (11)–(15) and (30). Sentence (iii) includes a new disjunctive term corresponding to rule (29); it is equivalent to (31). Sentence (iv) is the same logically valid formula as before. Sentence (i) includes a new disjunctive term corresponding to rule (28). It is equivalent to the following modification of (35):

$$\begin{aligned} & Holds(f,s) \equiv \\ & \quad \exists xys_1[f = On(x,y) \wedge s = Result(Move(x,y),s_1) \\ & \qquad \wedge Poss(Move(x,y),s_1)] \\ & \quad \vee \exists xys_1[f = Damaged(x) \\ & \qquad \wedge s = Result(Move(x,y),s_1) \\ & \qquad \wedge Poss(Move(x,y),s_1) \\ & \qquad \wedge Holds(Damaged(x), Result(Move(x,y),s_1))] \\ & \quad \vee \exists as_1[s = Result(a,s_1) \wedge Inertial(f) \\ & \qquad \wedge Poss(a,s_1) \wedge Holds(f,s_1) \wedge Holds(f,s)] \\ & \quad \vee [Holds(f,s) \wedge \neg Inertial(f)] \\ & \quad \vee [Holds(f,s) \wedge \neg \exists as_1(Poss(a,s_1) \\ & \qquad \wedge s = Result(a,s_1))]. \end{aligned} \qquad (42)$$

Finally, sentence (ii) is the same before, and it is equivalent to the universal closure of (36).

Consequently *CBW′* is equivalent to the conjunction of the universal closures of formulas (11)–(15), (30), (31), (36) and (42).

The assertion of Proposition 3 follows from this fact in view of Lemmas 2 and 4 and the following two assertions, characterizing the two halves of the equivalence (42):

Lemma 7. *$(42)^{\subset}$ is universally equivalent to (1).*

Lemma 8. *In the presence of (15), formula $(42)^{\supset}$ is universally equivalent to (32).*

Proof of Lemma 7. Similar to the proof of Lemma 1; $(42)^{\subset}$ can be written as the conjunction of five implications, and the implication containing *Damaged* is logically valid.

Proof of Lemma 8. The proof is similar to the proof of Lemma 3. After the first two simplification steps, we get:

$$\begin{array}{l} Holds(f,s) \land Inertial(f) \\ \land Poss(a,s_1) \land s = Result(a,s_1) \\ \quad \supset \exists xys_1[f = On(x,y) \\ \qquad \land s = Result(Move(x,y),s_1) \\ \qquad \land Poss(Move(x,y),s_1)] \\ \quad \lor \exists xys_1[f = Damaged(x) \\ \qquad \land s = Result(Move(x,y),s_1) \\ \qquad \land Poss(Move(x,y),s_1) \\ \qquad \land Holds(Damaged(x), Result(Move(x,y),s_1))] \\ \quad \lor \exists as_1[s = Result(a,s_1) \land Poss(a,s_1) \\ \qquad \land Holds(f,s_1)]. \end{array}$$

In the presence of (15), this is universally equivalent to

$$\begin{array}{l} Holds(f,s) \land Inertial(f) \\ \land Poss(a,s_1) \land s = Result(a,s_1) \\ \quad \supset \exists xy[f = On(x,y) \land a = Move(x,y) \\ \qquad \land Poss(Move(x,y),s_1)] \\ \quad \lor \exists xy[f = Damaged(x) \\ \qquad \land a = Move(x,y) \\ \qquad \land Poss(Move(x,y),s_1) \\ \qquad \land Holds(Damaged(x), Result(Move(x,y),s_1))] \\ \quad \lor [Poss(a,s_1) \land Holds(f,s_1)] \end{array}$$

and consequently to

$$\begin{array}{l} Holds(f,s) \land Inertial(f) \\ \land Poss(a,s_1) \land s = Result(a,s_1) \\ \quad \supset \exists xy[f = On(x,y) \land a = Move(x,y)] \\ \quad \lor \exists xy[f = Damaged(x) \land a = Move(x,y)] \\ \quad \lor Holds(f,s_1). \end{array}$$

This formula is universally equivalent to

$$\begin{array}{l} Holds(f, Result(a,s_1)) \land Inertial(f) \land Poss(a,s_1) \\ \quad \supset \exists xy[f = On(x,y) \land a = Move(x,y)] \\ \quad \lor \exists xy[f = Damaged(x) \land a = Move(x,y)] \\ \quad \lor Holds(f,s_1) \end{array}$$

and consequently to (32).

References

[Baker, 1991] Andrew Baker. Nonmonotonic reasoning in the framework of situation calculus. *Artificial Intelligence*, 49:5–23, 1991.

[Clark, 1978] Keith Clark. Negation as failure. In Herve Gallaire and Jack Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, New York, 1978.

[Geffner, 1990] Hector Geffner. Causal theories for nonmonotonic reasoning. In *Proc. AAAI-90*, pages 524–530, 1990.

[Giunchiglia and Lifschitz, 1998] Enrico Giunchiglia and Vladimir Lifschitz. An action language based on causal explanation. In *Proc. AAAI-98*, 1998. To appear.

[Kartha, 1993] Neelakantan Kartha. Soundness and completeness theorems for three formalizations of action. In *Proc. IJCAI-93*, pages 724–729, 1993.

[Lifschitz, 1991] Vladimir Lifschitz. Towards a metatheory of action. In James Allen, Richard Fikes, and Erik Sandewall, editors, *Proc. Second Int'l Conf. on Principles of Knowledge Representation and Reasoning*, pages 376–386, 1991.

[Lifschitz, 1994] Vladimir Lifschitz. Circumscription. In D.M. Gabbay, C.J. Hogger, and J.A. Robinson, editors, *The Handbook of Logic in AI and Logic Programming*, volume 3, pages 298–352. Oxford University Press, 1994.

[Lifschitz, 1997] Vladimir Lifschitz. On the logic of causal explanation. *Artificial Intelligence*, 96:451–465, 1997.

[Lin and Reiter, 1994] Fangzhen Lin and Raymond Reiter. State constraints revisited. *Journal of Logic and Computation*, 4:655–678, 1994.

[Lin, 1995] Fangzhen Lin. Embracing causality in specifying the indirect effects of actions. In *Proc. IJCAI-95*, pages 1985–1991, 1995.

[Lloyd and Topor, 1984] John Lloyd and Rodney Topor. Making Prolog more expressive. *Journal of Logic Programming*, 3:225–240, 1984.

[McCain and Turner, 1995] Norman McCain and Hudson Turner. A causal theory of ramifications and qualifications. In *Proc. IJCAI-95*, pages 1978–1984, 1995.

[McCain and Turner, 1997] Norman McCain and Hudson Turner. Causal theories of action and change. In *Proc. AAAI-97*, pages 460–465, 1997.

[McCarthy and Hayes, 1969] John McCarthy and Patrick Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 4, pages 463–502. Edinburgh University Press, Edinburgh, 1969. Reproduced in [McCarthy, 1990].

[McCarthy, 1990] John McCarthy. *Formalizing Common Sense: Papers by John McCarthy.* Ablex, Norwood, NJ, 1990.

[Reiter, 1991] Raymond Reiter. The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression.

In Vladimir Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 359–380. Academic Press, 1991.

[Schubert, 1990] Lenhart Schubert. Monotonic solution of the frame problem in the situation calculus: an efficient method for worlds with fully specified actions. In H.E. Kyburg, R. Loui, and G. Carlson, editors, *Knowledge Representation and Defeasible Reasoning*, pages 23–67. Kluwer, 1990.

[Turner, 1998] Hudson Turner. A logic of universal causation. *Artificial Intelligence*, 1998. To appear.

Sequential, Temporal GOLOG

Ray Reiter
Department of Computer Science
University of Toronto
Toronto, Canada, M5S 1A4
reiter@cs.toronto.edu

Abstract

We extend the ontology and foundational axioms of the sequential situation calculus to include time. When combined with a view of actions with durations as processes that are initiated and terminated by instantaneous actions, this explicit representation of time yields a very rich account of interleaving concurrency in the situation calculus. Based upon this axiomatization, we extend the semantics and interpreter for the situation calculus-based programming language GOLOG to the temporal domain, and illustrate the resulting increased functionality of the language via a GOLOG program describing the temporal behaviour of a coffee delivery robot. Among other features, this program illustrates how, in the GOLOG framework, one can represent concurrent processes with explicit time.

1 Introduction

The situation calculus [15] has long been the formalism of choice in artificial intelligence for theoretical investigations of properties of actions (e.g. [12, 1, 4]), but until recently, it has not been taken seriously as a specification or implementation language for practical problems in dynamic world modeling. Exceptions to this are the situation calculus-based programming languages GOLOG and CONGOLOG [11, 2], and some of their applications to planning [10, 20], robotics [7, 5, 3] and agent programming [8, 6]. The perspective being pursued by the Cognitive Robotics Group at the University of Toronto is to reduce the "traditional" reliance on *planning* for eliciting interesting robot behaviors, and instead provide the robot with *programs* written in a suitable high-level language [9], in our case, GOLOG or CONGOLOG. Such programming languages must be very expressive, providing a range of primitives for describing agent behaviors in complex worlds, for example, sensing actions, time, inter-agent communication, beliefs, goals, intentions, etc. Moreover, because of their complexity, they must do so in a semantically clear way. As indicated above, our approach to the design of such languages has been through the situation calculus.

The purpose of this paper is to extend the functionality of these languages by endowing them with the ability to represent time explicitly. Specifically, we extend the ontology and foundational axioms of the sequential situation calculus of [19] to include time. By pursuing an idea first proposed for the situation calculus by Pinto [17] and Ternovskaia [24], we show how one can view actions with durations as processes that are initiated and terminated by instantaneous actions. This conceptual shift, when coupled with an explicit representation for time, provides a rich account of interleaving concurrency in the situation calculus. Based upon the axioms for the sequential, temporal situation calculus, we next extend the semantics and interpreter for the situation calculus-based programming language GOLOG to the temporal domain. Finally, we illustrate the resulting increased functionality of the language with a GOLOG program describing the temporal behavior of a coffee delivery robot. Among other features, this program illustrates how, in the GOLOG framework, one can straightforwardly represent concurrent processes with explicit time.

2 The Situation Calculus

The situation calculus is a second order language specifically designed for representing dynamically changing worlds. All changes to the world are the result of named *actions*. A possible world history, which is simply a sequence of actions, is represented by a first order term called a *situation*. The constant S_0 is used to denote the *initial situation*, namely the empty history. Non-empty histories are constructed using a distinguished binary function symbol *do*; $do(\alpha, s)$ denotes the successor situation to s resulting from performing the action α. Actions may be parameterized. For example, $put(x, y)$ might stand for the action of putting object x on object y, in which case $do(put(A, B), s)$ denotes that situation resulting from placing A on B when the history is s. In the situation calculus, actions are denoted by first order terms, and situations (world histories) are also first order terms. For example, $do(putdown(A), do(walk(L), do(pickup(A), S_0)))$ is the situation denoting the world history consisting of the sequence of actions [pickup(A), walk(L), putdown(A)]. Notice that the sequence of actions in a history, in the order in which they occur, is obtained from a situation term by reading off the actions from right to left.

Relations whose truth values vary from situation to situation are called *relational fluents*. They are denoted by predicate symbols taking a situation term as their last argument. Similarly, functions whose values vary from situation to situation are called *functional fluents*, and are denoted by function symbols taking a situation term as their last argument. For example, $isCarrying(robot, p, s)$, meaning that a *robot* is carrying package p in situation s, is a relational fluent; $location(robot, s)$, denoting the location of *robot* in situation s, is a functional fluent.

To characterize the domain of situations, various foundational axioms have been proposed for the situation calculus [22, 13, 18]. The following set of axioms modifies these earlier proposals, and appears to be the simplest appropriate formulation of foundational axioms for the situation calculus [19]:

$$(\forall P).P(S_0) \wedge (\forall a, s)[P(s) \supset P(do(a, s))] \supset (\forall s)P(s), \qquad (1)$$

$$do(a, s) = do(a', s') \supset a = a' \wedge s = s',[1] \qquad (2)$$

$$\neg s \sqsubset S_0, \qquad (3)$$

$$s \sqsubset do(a, s') \equiv s \sqsubseteq s'. \qquad (4)$$

The first is a second order induction axiom. The next is a unique names axiom for situations. The last two axioms define an ordering relation $\sqsubset$ on situations. (Here, $s \sqsubseteq s'$ is an abbreviation for $s \sqsubset s' \vee s = s'$.) The intuitive reading of a situation is as a finite sequence of actions, in which case $s \sqsubseteq s'$ means that situation s' can be obtained from s by adding a finite number of actions onto s. Here, S_0 is a distinguished constant symbol of the language of the situation calculus, denoting the initial situation.[2]

[1]In what follows, lower case Roman characters will denote variables in formulas. Moreover, free variables will always be implicitly universally prenex quantified.

[2]Notice that in the situation calculus, the constant S_0 is just like NIL in the programming language LISP, and *do* acts like *cons*. Situations are simply *lists* of primitive actions. For example, the situation term $do(C, do(B, do(A, S_0)))$ is simply an alternative syntax for the LISP list $(C B A)$ $(= cons(C, cons(B, cons(A, NIL))))$. To obtain the action *history* corresponding to this term, namely the performance of action A, followed by B, followed by C, read this list from right to left. Therefore, when situation terms are read from right to left, the relation $s \sqsubset s'$ means that situation s is a proper subhistory of the situation s'. The induction axiom (1) simply provides induction for lists: If the empty list has property P and if, whenever list s has property P so does $cons(a, s)$, then all lists have property P.

According to these axioms, a situation is a finite sequence of actions. There are no constraints on the actions entering into such a sequence, so that it may not be possible to actually execute these actions one after the other. Frequently, we shall be interested only in *executable* situations, namely, those action sequences in which it is actually possible to perform the actions one after the other. To characterize such situations, we rely on the distinguished predicate symbol *Poss* of the language of the situation calculus. Intuitively, $Poss(a, s)$ means that it is possible to perform the action a in situation s. In specifying a domain of application, one would include axioms characterizing *Poss* for each primitive action of the domain. Now we can introduce the abbreviation:

$$s < s' \overset{def}{=} s \sqsubset s' \wedge (\forall a, s^*).s \sqsubset do(a, s^*) \sqsubseteq s' \supset Poss(a, s^*). \qquad (5)$$

So $s < s'$ means that s is an initial subsequence of s', and all the actions of s' following those of s can be executed one after the other. In particular, $S_0 < s$ means that every action in s is possible, so they can all be executed in sequence. To captures formally this intuitive concept of the executable situations, we introduce the abbreviation:

$$executable(s) \overset{def}{=} S_0 < s \vee S_0 = s.$$

In addition to the above domain independent axioms, one must specify other axioms when formalizing an application domain (details in [11]):

- *Action precondition axioms*, one for each primitive action, characterizing the relation *Poss*.
- *Successor state axioms*, one for each fluent. These capture the causal laws of the domain, together with a solution to the frame problem [21]. The solution to the frame problem embodied in these axioms applies only when all the primitive actions of the application domain are deterministic. Moreover, [21] does not treat state constraints, and therefore, does not address the ramification or qualification problems.[3]
- Unique names axioms for the primitive actions.
- Axioms describing the initial situation – what is true initially, before any actions have occurred. This is any finite set of sentences mentioning no situation term, or only the situation term S_0.

3 GOLOG

GOLOG [11] is a situation calculus-based logic programming language for defining complex actions us-

[3]But see [13, 16, 17] for possible ways to do this, while preserving the successor state axiom approach.

ing a repertoire of user specified primitive actions. GOLOG provides the usual kinds of imperative programming language control structures as well as three flavours of nondeterministic choice:

1. *Sequence:* α ; β. Do action α, followed by action β.
2. *Test actions:* p? Test the truth value of expression p in the current situation.
3. *While loops:* **while** p **do** α **endWhile.**
4. *Conditionals:* **if** p **then** α **else** β.
5. *Nondeterministic action choice:* $\alpha \mid \beta$. Do α or β.
6. *Nondeterministic choice of arguments:* $(\pi\ x)\alpha$. Nondeterministically pick a value for x, and for that value of x, do the action α.
7. *Nondeterministic repetition:* α^*. Do α a nondeterministic number of times.
8. *Procedures*, including recursion.

The semantics of GOLOG programs is defined by macro-expansion, using a ternary relation *Do* (see [11] for a full description). *Do* is defined inductively on the structure of its first argument as follows:

Primitive actions:

$$Do(a, s, s') \overset{def}{=} Poss(a, s) \wedge s' = do(a, s). \qquad (6)$$

Test actions:

$$Do(\phi?, s, s') \overset{def}{=} \phi[s] \wedge s = s'.$$

Here, ϕ is a test expression of our programming language; it stands for a situation calculus formula, but with all situation arguments suppressed. $\phi[s]$ denotes the situation calculus formula obtained by restoring situation variable s to all fluent names mentioned in ϕ.

Sequence:

$$Do(\delta_1; \delta_2, s, s') \overset{def}{=} (\exists s^*).Do(\delta_1, s, s^*) \wedge Do(\delta_2, s^*, s').$$

Nondeterministic choice of two actions:

$$Do(\delta_1 \mid \delta_2, s, s') \overset{def}{=} Do(\delta_1, s, s') \vee Do(\delta_2, s, s').$$

Nondeterministic choice of action arguments:

$$Do((\pi x)\ \delta(x), s, s') \overset{def}{=} (\exists x)\ Do(\delta(x), s, s').$$

Conditionals:

$$\textbf{if } p \textbf{ then } \alpha \textbf{ else } \beta \overset{def}{=} p? ; \alpha \mid \neg p? ; \beta.$$

Similar (but more complicated) definitions are given for iteration and procedures.

Do(*program*, s, s') is an *abbreviation* for a situation calculus formula whose intuitive reading is that s' is one of the situations reached from s by executing *program*. This means that to execute *program*, one must *prove*, using the situation calculus axiomatization of some background domain, the situation calculus formula $(\exists s)Do(program, S_0, s)$. Any binding for s obtained by a constructive proof of this sentence is an execution trace of *program*.

In [11] a GOLOG interpreter was given, written in Prolog. We present a variant of this here because we shall be suitably modifying it to accommodate time, and because we shall be presenting an example of a corresponding temporal GOLOG program.

A Golog Interpreter in Prolog

```
:- op(950, xfy, [:]). /* Sequence.*/
:- op(950, xfy, [#]). /* Nondeterministic action
                         choice.*/
:- op(800, xfy, [&]).   /* Conjunction */
:- op(850, xfy, [v]).   /* Disjunction */
:- op(870, xfy, [=>]).  /* Implication */
:- op(880,xfy, [<=>]).  /* Equivalence */

do(E1 : E2,S,S1) :- do(E1,S,S2), do(E2,S2,S1).
do(?(P),S,S) :- holds(P,S).
do(E1 # E2,S,S1) :- do(E1,S,S1) ; do(E2,S,S1).
do(if(P,E1,E2),S,S1) :-
            do(?(P) : E1 # ?(-P) : E2,S,S1).
do(star(E),S,S1) :- S1 = S ;  do(E : star(E),S,S1).
do(while(P,E),S,S1):-
                  do(star(?(P) : E) : ?(-P),S,S1).
do(pi(V,E),S,S1) :- sub(V,_,E,E1), do(E1,S,S1).
do(E,S,S1) :- proc(E,E1), do(E1,S,S1).
do(E,S,do(E,S)) :- primitive_action(E), poss(E,S).

/* sub(Name,New,T1,T2): T2 is T1 with Name
   replaced by New. */

sub(X1,X2,T1,T2) :- var(T1), T2 = T1.
sub(X1,X2,T1,T2) :- not var(T1), T1 = X1, T2 = X2.
sub(X1,X2,T1,T2) :- not T1 = X1, T1 =..[F|L1],
         sub_list(X1,X2,L1,L2), T2 =..[F|L2].
sub_list(X1,X2,[],[]).
sub_list(X1,X2,[T1|L1],[T2|L2]) :-
       sub(X1,X2,T1,T2), sub_list(X1,X2,L1,L2).

/* The holds predicate implements the Lloyd-Topor
   transformations on test conditions.  */

holds(P & Q,S) :- holds(P,S), holds(Q,S).
holds(P v Q,S) :- holds(P,S); holds(Q,S).
holds(P => Q,S) :- holds(-P v Q,S).
holds(P <=> Q,S) :- holds((P => Q) & (Q => P),S).
holds(-(-P),S) :- holds(P,S).
holds(-(P & Q),S) :- holds(-P v -Q,S).
holds(-(P v Q),S) :- holds(-P & -Q,S).
holds(-(P => Q),S) :- holds(-(-P v Q),S).
holds(-(P <=> Q),S) :-
      holds(-((P => Q) & (Q => P)),S).
holds(-all(V,P),S) :- holds(some(V,-P),S).
     /* Negation by failure */
holds(-some(V,P),S) :- not holds(some(V,P),S).
holds(-P,S) :- isAtom(P), not holds(P,S).
holds(all(V,P),S) :- holds(-some(V,-P),S).
holds(some(V,P),S) :- sub(V,_,P,P1), holds(P1,S).

/* The following clause treats the holds predicate
   for all atoms, including Prolog system
   predicates. For this to work properly, the
   GOLOG programmer must provide, for all atoms
   taking a situation argument, a clause giving
   the result of restoring its suppressed
   situation argument, for example:
     restoreSitArg(ontable(X),S,ontable(X,S)). */

holds(A,S) :- restoreSitArg(A,S,F), F ;
       not restoreSitArg(A,S,F), isAtom(A), A.

isAtom(A) :- not (A = -W ; A = (W1 & W2) ;
    A = (W1 => W2) ; A = (W1 <=> W2) ;
    A = (W1 v W2) ; A = some(X,W) ; A = all(X,W)).

restoreSitArg(poss(A),S,poss(A,S)).
```

It is possible to prove the soundness of this interpreter, with respect to the above semantics, when the usual Prolog closed world assumption is made about the initial database, namely, that complete information is available about the initial situation. Therefore, this interpreter is only suitable for applications in which this closed world assumption holds.[4] The `holds` predicate in this interpreter evaluates test conditions of GOLOG programs. Since such test conditions can be arbitrary first order formulas, the `holds` predicate first converts them to Prolog executable form using the Lloyd-Topor transformations [14].

4 Interleaving Concurrency in the Situation Calculus

The possibility of concurrent execution of actions leads to many difficult formal and conceptual problems, quite independently of the underlying knowledge representation language. For example, what can one mean by the concurrent action $\{walk(A, B), chewGum\}$? Intuitively, both actions have durations. By this concurrent action, do we mean that both actions have the same duration? That the time segment occupied by one is entirely contained in that occupied by the other? That their time segments merely overlap? What if there are three actions and the first overlaps the second, the second overlaps the third, but the first and third do not overlap; do they all occur concurrently? A representational device in the situation calculus for overcoming these problems is to conceive of such actions as *processes*, represented by relational fluents, and to introduce durationless (instantaneous) actions that initiate and terminate these processes [17, 24]. For example, instead of the monolithic action representation $walk(x, y)$, we might have instantaneous actions $startWalk(x, y)$ and $endWalk(x, y)$, and the process of walking from x to y, represented by the relational fluent $walking(x, y, s)$. $startWalk(x, y)$ causes the fluent $walking$ to become true, $endWalk(x, y)$ causes it to become false. Similarly, we might represent the $chewGum$ action by the pair of instantaneous actions $startChewGum$ and $endChewGum$, and the relational fluent $chewingGum(s)$. It is straightforward to represent these fluents and instantaneous actions in the situation calculus. For example, here are the action precondition and successor state axioms for the walking action:

$$Poss(startWalk(x, y), s) \equiv \\ \neg(\exists u, v)walking(u, v, s) \wedge location(s) = x,$$

$$Poss(endWalk(x, y), s) \equiv walking(x, y, s),$$

$$walking(x, y, do(a, s)) \equiv a = startWalk(x, y) \vee \\ walking(x, y, s) \wedge a \neq endWalk(x, y),$$

$$location(do(a, s)) = y \equiv (\exists x)a = endWalk(x, y) \vee \\ location(s) = y \wedge \neg(\exists x, y')a = endWalk(x, y').$$

With this device of instantaneous *start* and *end* actions in hand, we can represent arbitrarily complex concurrency. For example,

$$do(endWalk(A, B), do(endChewGum, \\ do(startChewGum, do(startWalk(A, B), S_0)))),$$

in which the gum-chewing process is initiated after the walking process, and terminated before the end of the walking process. Or, we can have the gum-chewing start before the walking, and terminate before the walking ends:

$$do(endWalk(A, B), do(endChewGum, \\ do(startWalk(A, B), do(startChewGum, S_0)))).$$

In other words, we can represent any overlapping occurrences of the walking and chewing gum processes, *except for exact co-occurrences of any of the instantaneous initiating and terminating actions.* For many applications, this is sufficient. The great advantage is that this style of interleaved concurrency can be represented in the "classical" sequential situation calculus, and no new extensions of the theory are necessary. However, as yet, we have no explicit representation for time; the axioms for the situation calculus capture a purely qualitative notion of time. Sequential action occurrence is the only temporal concept represented by the axioms; an action occurs *before* or *after* another. It may occur one millisecond or one year before its successor; the axioms are neutral on this question. So a situation $do(A_n, do(A_{n-1}, \ldots, do(A_1, S_0) \cdots))$ must be understood as a world history in which, after a nondeterministic period of time, A_1 occurs, then, after a nondeterministic period of time, A_2 occurs, etc. If, for example, this action sequence was the result of a GOLOG robot program execution, then the robot's action execution system would make the decision about the exact times at which these actions would be performed sequentially in the physical world, but the axioms, being silent on action occurrence times, con-

[4]Of course, the general theory of GOLOG does not suffer from this restriction, only the specific implementation given above.

tribute nothing to this decision. Our objective now is to show how to incorporate time into the situation calculus, after which one can specify axiomatically and via GOLOG programs the times at which actions are to occur.

5 The Sequential, Temporal Situation Calculus

Now, we add an explicit representation for time to the sequential situation calculus of Section 2. This will allow us to specify the exact times, or a range of times, at which actions in a world history occur. For the reasons indicated in the previous section, we consider only instantaneous actions. We want to represent the fact that a given such action occurs at a particular time. Recall that in the situation calculus, actions are denoted by first order terms, like $start\_meeting(Susan)$ or $bounce(ball, wall)$. Our proposal for adding a time dimension to the situation calculus is to provide a new temporal argument to all instantaneous actions, denoting the time at which that action occurs. Thus, $bounce(ball, wall, 7.3)$ might be the instantaneous action of $ball$ bouncing against $wall$ at time 7.3.

We now extend the foundational axioms for the sequential situation calculus of Section 2 to accommodate time. We retain these axioms, and add only one new axiom. Also, the definition of executable situation requires modification. We begin by introducing a function symbol $time$: $time(a)$ denotes the time of occurrence of action a. This means that in any application involving a particular action $A(\vec{x}, t)$, we shall need an axiom telling us the time of the action A: $time(A(\vec{x}, t)) = t$, for example, $time(start\_meeting(person, t)) = t$. Next, it will be convenient to have a function $start$: $start(s)$ denotes the start time of situation s. This requires the axiom:

$$start(do(a, s)) = time(a). \tag{7}$$

We do not define the start time of S_0; this is arbitrary, and may (or may not) be specified to be any number, depending on the application. Notice also that we imagine temporal variables to range over the reals, although nothing prevents them from ranging over the integers, rationals, or anything else on which a binary relation $<$ is defined. In this connection, we do not provide axioms for the reals (or integers), but rely instead on the standard interpretation of the reals and their operations (addition, multiplication, etc) and relations ($<$, $\leq$, etc).

Next we reconsider the relation $<$ on situations. Intuitively, $s \leq s'$ means that one can get to s' from s by a sequence of possible actions. Consider the situation $do(bounce(B, W, 4), do(start\_meeting(Susan, 6), S_0))$, in which the time of the second action precedes that of the first. Intuitively, we do not want to consider such an action sequence executable, and we amend our definition (5) for $<$ accordingly:

$$\begin{aligned} s < s' \overset{def}{=} & \; s \sqsubset s' \land \\ & (\forall a, s^*). s \sqsubset do(a, s^*) \sqsubseteq s' \supset \\ & \quad Poss(a, s^*) \land start(s^*) \leq time(a). \end{aligned} \tag{8}$$

Now, $s < s'$ means that one can get to s' from s by a sequence of possible actions, and moreover, the times of those action occurrences must be nondecreasing. We are here overloading the predicate $<$; it is used to order situations as well as numbers in the temporal domain. Its meaning will always be clear from context.

Finally, notice that the constraint $start(s^*) \leq time(a)$ in abbreviation (8) permits executable action sequences in which the time of an action may be the same as the time of a preceding action. For example,

$$\begin{aligned} & do(end\_lunch(Bill, 4), \\ & \qquad do(start\_meeting(Susan, 4), S_0)) \end{aligned}$$

might be a perfectly good executable situation, which is defined by a sequence of two actions, each of which has the same occurrence time, but one of which ($start\_meeting(Susan, 4)$) "occurs before" the other ($end\_lunch(Bill, 4)$). This means that we provide for concurrent execution of instantaneous actions, but unlike the true concurrency treated in [23], we are here giving an *interleaving* account of concurrency. There are many reasons for allowing two or more interleaved actions to have the same occurrence times. One is we can often give an interleaving account of action co-occurrences without introducing the more complex formal machinery of [23]. Another is that often an action occurrence serves as an enabling condition for the simultaneous occurrence of another action. For example, cutting a weighted string at time t enables the action $startFalling(t)$. Both actions occur at the same time, but conceptually, the falling event happens "immediately after" the cutting. Accordingly, we want to treat the situation $do(startFalling(t), do(cutString(t), S_0))$ as an executable situation.

There are many advantages to using interleaving instead of true concurrency, whenever this is possible. For example, the precondition interaction problem [17] cannot arise in this case, neither can the possibility of infinitely many action co-occurrences [23]. Moreover, as we shall see with the example to follow, the combination of instantaneous actions, explicit representation of time, and an interleaving account of concurrency provides for a very rich, yet formally simple representation language for processes that overlap in temporally complex ways.

(1) - (4) and (7) are the foundational axioms for the *sequential, temporal situation calculus.* The development given above of these foundational axioms has many similarities to that given by Reiter in [23] for the *concurrent, temporal situation calculus.* The principal difference is that [23] treats true concurrency,

where concurrent actions are sets of primitive instantaneous actions. It is possible to obtain the foundational axioms for the sequential, temporal situation calculus from those of [23] by requiring that all concurrent actions be singleton sets, and identifying the primitive action A with the "concurrent" action $\{A\}$, but the above approach, where we started from scratch, seemed to us conceptually more attractive.

6 Sequential, Temporal GOLOG

With the above axioms for the sequential, temporal situation calculus in hand, it is easy to modify the GOLOG semantics and interpreter to accommodate time. Semantically, we need only change the definition of the Do macro for primitive actions (6) to:

$$Do(a, s, s') \stackrel{def}{=} Poss(a, s) \wedge start(s) \leq time(a) \wedge s' = do(a, s).$$

Everything else about the definition of Do remains the same. One can prove that with this modification, whenever a sequential temporal GOLOG program terminates, it does so in an executable situation according to the revised definition (8) for executable situations. To suitably modify the GOLOG interpreter of Section 3, replace the clause

```
do(E,S,do(E,s)):- primitive_action(E),
                  poss(E,S).
```

by

```
do(E,S,do(E,S)) :- primitive_action(E),
                   poss(E,S), start(S,T1),
                   time(E,T2), T1 <= T2.
```

Finally, because we have introduced a new predicate, *start*, taking a situation argument, we must also augment the earlier GOLOG interpreter with the clause:

```
start(do(A,S),T) :- time(A,T).
```

We can now write sequential, temporal GOLOG programs. However, to execute such programs, the GOLOG interpreter must have a temporal reasoning component. It must, for example, be able to infer that $T_1 = T_2$ when given that $T_1 \leq T_2 \wedge T_2 \leq T_1$. While such a special purpose temporal theorem prover could be written and included in the GOLOG interpreter, we prefer to rely on a logic programming language with a built-in constraint solving capability. Specifically, we appeal to the ECRC Common Logic Programming System ECLIPSE 3.5.2; this provides a built-in Simplex algorithm for solving linear equations and inequalities over the reals. So we shall assume that our GOLOG program makes use of linear temporal relations like $2 * T_1 + T_2 = 5$ and $3 * T_2 - 5 \leq 2 * T_3$, and rely on ECLIPSE to perform the temporal reasoning for us. ECLIPSE provides a special syntax for those relations over the reals for which it provides a built-in theorem prover. These are: $ =, $ <>, $ >=, $ >, $ <=, $ <, with the obvious meanings. So, in ECLIPSE, the above modification of the GOLOG interpreter to include time is:

```
do(E,S,do(E,S)) :- primitive_action(E),
                   poss(E,S), start(S,T1),
                   time(E,T2), T1 $<= T2.
```

All other clauses of the earlier interpreter as given in Section 3 will work correctly under ECLIPSE.

6.1 Example: A Coffee Delivery Robot

Here, we describe a robot whose task is to deliver coffee in an office environment. The robot is given a schedule of every employee's preferred coffee periods, as well as information about the times it takes to travel between various locations in the office. The robot can carry just one cup of coffee at a time, and there is a central coffee machine from which it gets the coffee. Its task is to schedule coffee deliveries such that, if possible, everyone gets coffee during his/her preferred time periods. For simplicity, we assume that both the actions of picking up a cup of coffee, and giving it to someone are instantaneous. We represent the action of going from one location to another, which intuitively does have a duration, by a process and a pair of instantaneous *start* and *end* actions, as described in Section 4.

Primitive actions:

- $pickupCoffee(t)$. The robot picks up a cup of coffee from the coffee machine at time t.
- $giveCoffee(p, t)$. The robot gives a cup of coffee to p at time t.
- $startGo(loc_1, loc_2, t)$. The robot starts to go from location loc_1 to loc_2 at time t.
- $endGo(loc_1, loc_2, t)$. The robot ends its process of going from location loc_1 to loc_2 at time t.

Fluents:

- $robotLocation(s)$. A functional fluent denoting the robot's location in situation s.
- $hasCoffee(person, s)$. $person$ has coffee in s.
- $going(loc_1, loc_2, s)$. In situation s, the robot is going from loc_1 to loc_2.
- $holdingCoffee(s)$. In situation s, the robot is holding a cup of coffee.

Situation Independent Predicates and Functions

- $wantsCoffee(person, t_1, t_2)$. $person$ wants to receive coffee at some point in the time period $[t_1, t_2]$.
- $office(person)$. Denotes the office of $person$.
- $travelTime(loc_1, loc_2)$. Denotes the amount of time that the robot takes to travel between loc_1 and loc_2.
- CM. Constant denoting coffee machine's location.

- *Sue*, *Mary*, *Bill*, *Joe*. Constants denoting people.

Action Precondition Axioms:

$$Poss(pickupCoffee(t), s) \equiv \neg holdingCoffee(s) \wedge robotLocation(s) = CM,$$

$$Poss(giveCoffee(person, t), s) \equiv holdingCoffee(s) \wedge robotLocation(s) = office(person),$$

$$Poss(startGo(loc_1, loc_2, t), s) \equiv \neg(\exists l, l')going(l, l', s) \wedge loc_1 \neq loc_2 \wedge robotLocation(s) = loc_1,$$

$$Poss(endGo(loc_1, loc_2, t), s) \equiv going(loc_1, loc_2, s).$$

Successor State Axioms

$$hasCoffee(person, do(a, s)) \equiv (\exists t)a = giveCoffee(person, t) \vee hasCoffee(person, s),$$

$$robotLocation(do(a, s)) = loc \equiv (\exists t, loc')a = endGo(loc', loc, t) \vee robotLocation(s) = loc \wedge \neg(\exists t, loc', loc'')a = endGo(loc', loc'', t),$$

$$going(l, l', do(a, s)) \equiv (\exists t)a = startGo(l, l', t) \vee going(l, l', s) \wedge \neg(\exists t)a = endGo(l, l', t),$$

$$holdingCoffee(do(a, s)) \equiv (\exists t)a = pickupCoffee(t) \vee holdingCoffee(s) \wedge \neg(\exists person, t)a = giveCoffee(person, t).$$

Initial Situation

Unique names axioms stating that the following terms are pairwise unequal:

$$Sue,\ Mary,\ Bill,\ Joe,\ CM,\ office(Sue),\ office(Mary),\ office(Bill),\ office(Joe).$$

Initial Fluent values:

$$robotLocation(S_0) = CM,\quad start(S_0) = 0,\quad \neg(\exists p)hasCoffee(p, S_0), \neg holdingCoffee(S_0),\quad \neg(\exists l, l')going(l, l', S_0).$$

Coffee delivery preferences. The following expresses that $(Sue, 140, 160), \ldots, (Joe, 90, 100)$ are all, and only, the tuples in the *wantsCoffee* relation.

$$\begin{aligned} wantsCoffee(p, t_1, t_2) \equiv\ & p = Sue \wedge t_1 = 140 \wedge t_2 = 160 \vee \\ & p = Mary \wedge t_1 = 130 \wedge t_2 = 170 \vee \\ & p = Bill \wedge t_1 = 100 \wedge t_2 = 110 \vee \\ & p = Joe \wedge t_1 = 90 \wedge t_2 = 100. \end{aligned}$$

Robot travel times:

$$\begin{aligned} & travelTime(CM, office(Sue)) = 15, \\ & travelTime(CM, office(Mary)) = 10, \\ & travelTime(CM, office(Bill)) = 8, \\ & travelTime(CM, office(Joe)) = 10. \\ & travelTime(l, l') = travelTime(l', l), \\ & travelTime(l, l) = 0. \end{aligned}$$

Action Occurrence Times:

$$\begin{aligned} & time(pickupCoffee(t)) = t, \\ & time(giveCoffee(person, t)) = t, \\ & time(startGo(loc_1, loc_2, t)) = t, \\ & time(endGo(loc_1, loc_2, t)) = t. \end{aligned}$$

GOLOG Procedures

```
proc deliverCoffee(t)      % Beginning at time t
                 % the robot serves coffee to everyone,
                 % if possible. Else the program fails.
  $now \leq t?$ ;
  {[$(\forall p, t', t'').wantsCoffee(p, t', t'') \supset hasCoffee(p)$]?
   |
   if $robotLocation = CM$ then $deliverOneCoffee(t)$
      else $goto(CM, t)$ ; $deliverOneCoffee(now)$
   endIf}
endProc
```

The above procedure introduces a functional fluent *now(s)*, which is identical to the fluent *start(s)*. We use it instead of *start* because it has a certain mnemonic value, but, like *start*, it denotes the *current time*.

```
proc deliverOneCoffee(t)     % Assuming the robot
                     % is at the coffee machine,
                     % it delivers one cup of coffee.
  $(\pi p, t_1, t_2, wait)$[{$wantsCoffee(p, t_1, t_2) \wedge$
                  $\neg hasCoffee(p) \wedge wait \geq 0 \wedge$
                  $t_1 \leq t + wait +$
                      $travelTime(CM, office(p))$
                    $\leq t_2$}? ;
                 $pickupCoffee(t + wait)$ ;
                 $goto(office(p), now)$ ;
                 $giveCoffee(p, now)$ ;
                 $deliverCoffee(now)$]
endProc

proc goto(loc, t)       % Beginning at time t the
                        % robot goes to loc.
  $goBetween(robotLocation, loc,$
                 $travelTime(robotLocation, loc), t)$
endProc

proc goBetween(loc1, loc2, $\Delta$, t)      % Beginning at
        % time t the robot goes from loc1 to loc2,
        % taking $\Delta$ time units for the transition.
  $startGo(loc1, loc2, t)$ ; $endGo(loc1, loc2, t + \Delta)$
endProc
```

The following sequential temporal GOLOG program implements the above specification.

Sequential Temporal GOLOG Program for a Coffee Delivery Robot

```
/* GOLOG Procedures */

proc(deliverCoffee(T),
  ?(some(t, now(t) & t $<= T)) :
  (?(all(p,all(t1,all(t2,wantsCoffee(p,t1,t2) =>
                          hasCoffee(p)))))
   #
   pi(rloc, ?(robotLocation(rloc)) :
          if(rloc = cm,
              /* THEN */
                 deliverOneCoffee(T),
              /* ELSE */
                 goto(cm,T) : pi(t, ?(now(t)) :
                   deliverOneCoffee(t)))))).

proc(deliverOneCoffee(T),
  pi(p, pi(t1, pi(t2, pi(wait, pi(travTime,
     ?(wantsCoffee(p,t1,t2) & -hasCoffee(p) &
       wait $>= 0 &
       travelTime(cm,office(p),travTime) &
       t1 $<= T + wait + travTime & T + wait +
              travTime $<= t2) :
     pi(t, ?(t $= T + wait) : pickupCoffee(t)) :
     pi(t, ?(now(t)) : goto(office(p),t)) :
     pi(t, ?(now(t)) : giveCoffee(p,t)) :
     pi(t, ?(now(t)) : deliverCoffee(t)))))))).


proc(goto(L,T),
  pi(rloc,?(robotLocation(rloc)) :
    pi(deltat,?(travelTime(rloc,L,deltat)) :
      goBetween(rloc,L,deltat,T)))).

proc(goBetween(Loc1,Loc2,Delta,T),
   startGo(Loc1,Loc2,T) :
   pi(t, ?(t $= T + Delta) : endGo(Loc1,Loc2,t))).

/* Preconditions for Primitive Actions */

poss(pickupCoffee(T),S) :- not holdingCoffee(S),
                           robotLocation(cm,S).

poss(giveCoffee(Person,T),S) :- holdingCoffee(S),
                 robotLocation(office(Person),S).

poss(startGo(Loc1,Loc2,T),S) :- not going(L,LL,S),
          not Loc1 = Loc2, robotLocation(Loc1,S).

poss(endGo(Loc1,Loc2,T),S) :- going(Loc1,Loc2,S).

/* Successor State Axioms */

hasCoffee(Person,do(A,S)) :-
  A = giveCoffee(Person,T) ; hasCoffee(Person,S).

robotLocation(Loc,do(A,S)) :- A=endGo(Loc1,Loc,T) ;
  robotLocation(Loc,S), not A=endGo(Loc2,Loc3,T).

going(Loc1,Loc2,do(A,S)) :- A=startGo(Loc1,Loc2,T) ;
   going(Loc1,Loc2,S), not A = endGo(Loc1,Loc2,T).

holdingCoffee(do(A,S)) :- A = pickupCoffee(T) ;
   holdingCoffee(S), not A = giveCoffee(Person,T).

/* Initial Situation */

robotLocation(cm,s0).   start(s0,0).
wantsCoffee(sue,140,160).
wantsCoffee(bill,100,110).
wantsCoffee(joe,90,100).
wantsCoffee(mary,130,170).
travelTime0(cm,office(sue),15).
travelTime0(cm,office(mary),10).
travelTime0(cm,office(bill),8).
travelTime0(cm,office(joe),10).
travelTime(L,L,0).
travelTime(L1,L2,T) :- travelTime0(L1,L2,T) ;
                       travelTime0(L2,L1,T).

/* Action occurrence time is its last argument. */

time(pickupCoffee(T),T).
time(endGo(Loc1,Loc2,T),T).
time(giveCoffee(Person,T),T).
time(startGo(Loc1,Loc2,T),T).

/* Restore situation arguments to fluents. */

restoreSitArg(robotLocation(Rloc),S,
              robotLocation(Rloc,S)).
restoreSitArg(hasCoffee(Person),S,
              hasCoffee(Person,S)).
restoreSitArg(going(Loc1,Loc2),S,
              going(Loc1,Loc2,S)).
restoreSitArg(holdingCoffee,S,
              holdingCoffee(S)).

/* Primitive Action Declarations */

primitive_action(pickupCoffee(T)).
primitive_action(giveCoffee(Person,T)).
primitive_action(startGo(Loc1,Loc2,T)).
primitive_action(endGo(Loc1,Loc2,T)).

/* Fix on a solution to the temporal
   constraints. */

chooseTimes(s0).
chooseTimes(do(A,S)) :- chooseTimes(S),
                        time(A,T), rmin(T).

/* "now" is a synonym for "start".  */

now(S,T) :- start(S,T).
restoreSitArg(now(T),S,now(S,T)).
```

A problem with our constraint logic programming approach to coffee delivery is that the execution of the GOLOG call `do(deliverCoffee(1),s0,S)` will not, in general, result in a fully instantiated sequence of actions `S`. The actions in that sequence will not have their occurrence times uniquely determined; rather, these occurrence times will consist of all feasible solutions to the system of constraints generated by the program execution. So, to get a fixed schedule of coffee delivery, we must determine one or more of these feasible solutions. The relation `chooseTimes(S)` in the above program does just that. Beginning with the first action in the situation history, `S`, `chooseTimes` determines the time of that action (which, in general, will be a Prolog variable since the ECLIPSE constraint solver will not have determined a unique value for that action's occurrence time). It then minimizes (via `rmin(T)`) that time, relative to the current set of temporal constraints generated by executing the coffee delivery program. Then, having fixed the occurrence time of the first action, it repeats with the second action, etc. In this way, chooseTimes selects a particular solution to the linear temporal constraints generated by the program, thereby producing one of many possible schedules for the robot. Of course, there is nothing special about chooseTimes; any method for obtaining one or more solutions to the constraints would do just as well.

The following is the output obtained from this program under the temporal GOLOG interpreter of Section 6. We use the relation chooseTimes(S) to select a solution to the temporal constraints.

```
[eclipse 2]: do(deliverCoffee(1),s0,S),
             chooseTimes(S).

S = do(giveCoffee(mary,165),do(endGo(cm,
office(mary),165),do(startGo(cm,office(mary),155),
do(pickupCoffee(155),do(endGo(office(sue),cm,155),
do(startGo(office(sue), cm, 140), do(giveCoffee(
sue,140),do(endGo(cm,office(sue),140),do(startGo(
cm,office(sue),125),do(pickupCoffee(125),do(
endGo(office(bill),cm,116),do(startGo(office(bill),
cm,108),do(giveCoffee(bill,108),do(endGo(cm,
office(bill),108),do(startGo(cm,office(bill),100),
do(pickupCoffee(100),do(endGo(office(joe),cm,100),
do(startGo(office(joe),cm,90),do(giveCoffee(joe,
90),do(endGo(cm,office(joe),90),do(startGo(cm,
office(joe),80),do(pickupCoffee(80), s0))))))))))
)))))))))))      More? (;)
```

As is usual with GOLOG applications, the above computation would be done off line; as yet, the robot has not performed any physical actions in the world. To have it physically deliver coffee, the action sequence in the above execution trace would be passed to the robot's primitive execution module.

6.2 A Singing Robot

To simplify the exposition, we did not endow the above program with any interleaving execution of processes, as described in Section 4. This would, however, be easy to do. Suppose we wanted our robot to sing a song, but only while it is in transit between locations. Introduce two instantaneous actions $startSing(t)$ and $endSing(t)$, and a process fluent $singing(s)$, with action precondition and successor state axioms:

$$Poss(startSing(t), s) \equiv \neg singing(s).$$

$$Poss(endSing(t), s) \equiv singing(s),$$

$$singing(do(a, s)) \equiv (\exists t)a = startSing(t) \lor singing(s) \land \neg(\exists t)a = endSing(t).$$

Then the following version of the GOLOG procedure *goBetween* turns the robot into a singing waiter:

```
proc $goBetween(loc1, loc2, \Delta, t)$
  $startGo(loc1, loc2, t)$ ;
  $startSing(t)$ ; $endSing(t + \Delta)$ ;
  $endGo(loc1, loc2, t + \Delta)$
endProc
```

This provides a temporal, interleaving account of the concurrent execution of two processes: singing and moving between locations.

7 Conclusions and Future Work

We have extended the ontology and foundational axioms of the sequential situation calculus to include time, and showed how one can view actions with durations as processes that are initiated and terminated by instantaneous actions. This conceptual shift, when coupled with an explicit representation for time, provides a rich account of interleaving concurrency in the situation calculus. Based upon the axioms for the sequential, temporal situation calculus, we extended the semantics and interpreter for the situation calculus-based programming language GOLOG to the temporal domain. Finally, we illustrated the resulting increased functionality of the language with a GOLOG program describing the temporal behavior of a coffee delivery robot, including the concurrent processes of delivering coffee while singing a song.

CONGOLOG [2] is a much richer language than GOLOG, that includes facilities for interleaving concurrent execution, prioritized interrupts and exogenous actions. Augmenting CONGOLOG with a temporal component is a straightforward exercise, and can be done by changing its semantics and implementation in a manner exactly parallel to the approach of Section 6 for GOLOG.

Ongoing work along the lines of this paper includes controlling an RWI B21 autonomous robot to perform temporal scheduling tasks in an office environment. In this setting, it would be unrealistic to expect the robot to execute a schedule like that returned by our coffee delivery program. Frequently, it will be impossible to meet the exact times in such a schedule, for example, if the robot is unexpectedly delayed in traveling to the coffee machine. One approach we are exploring is to have the robot monitor its own execution, using the situation calculus-based execution monitor of [3], recomputing what remains of the schedule after it has determined (by sensing its internal clock) the actual occurrence times of its actions. We do not instantiate a schedule's action occurrence times (as we did using chooseTimes(S)), but leave these free, subject to the constraints generated by the GOLOG program. Whenever the robot physically performs an action, it senses the action's actual occurrence time, adds this to the constraints, then computes a remaining schedule, or fails if no continuing schedule can be found.

Acknowledgements:

This research was supported by grants from the Natural Sciences and Engineering Research Council of Canada, the Institute for Robotics and Intelligent Systems of the Government of Canada, and the Information Technology

Research Centre of the Government of Ontario. Mikhail Soutchanski provided valuable feedback and corrections for an earlier version of this paper.

References

[1] M. Gelfond, V. Lifschitz, and A. Rabinov. What are the limitations of the situation calculus? In *Working Notes, AAAI Spring Symposium Series on the Logical Formalization of Commonsense Reasoning*, pages 59–69, 1991.

[2] G. De Giacomo, Y. Lespérance, and H.J. Levesque. Reasoning about concurrent execution, prioritized interrupts, and exogenous actions in the situation calculus. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1221–1226, Nagoya, Japan, 1997.

[3] G. De Giacomo, R. Reiter, and M. Soutchanski. Execution monitoring of high-level robot programs. In A.G. Cohn and L.K. Schubert, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixth International Conference (KR'98)*. Morgan Kaufmann Publishers, San Francisco, CA, 1998.

[4] S. Hanks and D. McDermott. Default reasoning, nonmonotonic logics, and the frame problem. In *Proceedings of the National Conference on Artificial Intelligence (AAAI'86)*, pages 328–333, 1986.

[5] M. Jenkin, Y. Lespérance, H.J. Levesque, F. Lin, J. Lloyd, D. Marcu, R. Reiter, R.B. Scherl, and K. Tam. A logical approach to portable high-level robot programming. In *Proceedings of the Tenth Australian Joint Conference on Artificial Intelligence (AI'97)*, Perth, Australia, 1997. Invited paper.

[6] Y. Lespérance, H. Levesque, F. Lin, D. Marcu, R. Reiter, and R. Scherl. Foundations of a logical approach to agent programming. In M. Wooldridge, J.P. Muller, and M. Tambe, editors, *Intelligent Agents Vol. II - Proc. 1995 Workshop on Agent Theories, Architectures, and Languages (ATAL-95)*, pages 331–346. Springer-Verlag, Lecture Notes in Art. Intell., 1996.

[7] Y. Lespérance, H.J. Levesque, F. Lin, D. Marcu, R. Reiter, and R. Scherl. A logical approach to high-level robot programming - a progress report. In *Control of the Physical World by Intelligent Systems, Working Notes of the 1994 AAAI Fall Symp.*, 1994.

[8] Y. Lespérance, H.J. Levesque, and R. Reiter. A situation calculus approach to modeling and programming agents. In A. Rao and M. Wooldridge, editors, *Foundations and Theories of Rational Agency*, 1997. In press.

[9] H. L. Levesque and R. Reiter. High-level robotic control: beyond planning. Position paper. AAAI 1998 Spring Symposium: Integrating Robotics Research: Taking the Next Big Leap. Stanford University, March 23-25, 1998. http://www.cs.toronto.edu/~cogrobo/.

[10] H.J. Levesque. What is planning in the presence of sensing? In *Proceedings of the National Conference on Artificial Intelligence (AAAI'96)*, pages 1139–1146, 1996.

[11] H.J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. Scherl. GOLOG: a logic programming language for dynamic domains. *J. of Logic Programming, Special Issue on Actions*, 31(1-3):59–83, 1997.

[12] V. Lifschitz. Toward a metatheory of action. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 376–386, Los Altos, CA, 1991. Morgan Kaufmann Publishers, San Francisco, CA.

[13] F. Lin and R. Reiter. State constraints revisited. *J. of Logic and Computation, special issue on actions and processes*, 4:655–678, 1994.

[14] J.W. Lloyd. *Foundations of Logic Programming*. Springer Verlag, second edition, 1987.

[15] J. McCarthy. Situations, actions and causal laws. Technical report, Stanford University, 1963. Reprinted in Semantic Information Processing (M. Minsky ed.), MIT Press, Cambridge, Mass., 1968, pp. 410-417.

[16] Sheila A. McIlraith. *Towards a Formal Account of Diagnostic Problem Solving*. PhD thesis, Department of Computer Science, University of Toronto, Toronto, Ontario, Canada, 1997.

[17] J.A. Pinto. *Temporal Reasoning in the Situation Calculus*. PhD thesis, University of Toronto, Department of Computer Science, 1994.

[18] Javier Pinto. Occurrences and Narratives as Constraints in the Branching Structure of the Situation Calculus. Submitted to the Journal of Logic and Computation
URL = ftp://lyrcc.ing.puc.cl/pub/jpinto/jlc.ps.gz.

[19] F. Pirri and R. Reiter. Some contributions to the metatheory of the situation calculus. 1998. Submitted for publication.
http://www.cs.toronto.edu/~cogrobo/.

[20] R. Reiter. *Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems*. In preparation. Draft available at http://www.cs.toronto.edu/~cogrobo/.

[21] R. Reiter. The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression. In Vladimir Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 359–380. Academic Press, San Diego, CA, 1991.

[22] R. Reiter. Proving properties of states in the situation calculus. *Artificial Intelligence*, 64:337–351, 1993.

[23] R. Reiter. Natural actions, concurrency and continuous time in the situation calculus. In L.C. Aiello, J. Doyle, and S.C. Shapiro, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR'96)*, pages 2–13. Morgan Kaufmann Publishers, San Francisco, CA, 1996.

[24] E. Ternovskaia. Interval situation calculus. In *Proc. of ECAI'94 Workshop W5 on Logic and Change*, pages 153–164, Amsterdam, August 8-12, 1994.

Building Models of Prediction Theories

Graham White, John Bell and Wilfrid Hodges
Applied Logic Group
Department of Computer Science
and School of Mathematical Sciences
Queen Mary and Westfield College
University of London
London E1 4NS, U.K.
{graham, jb}@dcs.qmw.ac.uk, W.Hodges@qmw.ac.uk

Abstract

This paper is concerned with the implementation of Prediction Theories. Rather than developing and then attempting to implement a proof theory, it aims instead to implement their model theory by building computational counterparts of their relevant models. Prediction Theories are outlined and examples involving inertia, qualifications, ramifications and concurrency are given. The basic model-building algorithm is then presented and its completeness and correctness is proved. Several extensions to this algorithm are then outlined. Finally, the performance of the algorithms is analysed.

1 Introduction

This paper is concerned with the implementation of Prediction Theories (PTs) (Bell, 1997). PTs were developed to represent predictive common sense causal reasoning. In particular they provide a model-theoretic means of representing inertia, ramifications, qualifications, simultaneity, and non-determinism. Rather than developing and then attempting to implement a proof theory for PTs, this paper aims to implement their model theory, by showing how to construct the computational counterparts of the relevant models of a given PT. This results in a direct, appropriate and efficient implementation. It also provides a detailed example of the model-based theory of pragmatic reasoning outlined in (Bell, 1995). The work reported here represents a significant extension of the earlier example of the model-building implementation of the extension of Shoham's Causal Theories (Shoham, 1988; Bell, 1991; Bell, 1996). In particular, in (Bell, 1996) syntactic constraints ensure that there is a unique model of a given theory. Thus even reasonably simple cases of common sense causal reasoning—such as the combination of inertia and qualifications or inertia and context-dependent ramifications—could not be represented. Consequently, given uniqueness, the model-building task is relatively straightforward. By contrast, PTs can be used to represent complex examples of common sense causal reasoning, and the problem of developing a model-building algorithm is correspondingly difficult.

In order to make the paper self contained, an overview of PTs is given in the next section and examples involving inertia, qualifications, ramifications and concurrency are given. Model building is then discussed in Section 3. The basic model-building algorithm is given and its completeness and correctness is proved. Several extensions to the algorithm are then outlined. Finally, the performance of the algorithms is analysed. The full version of this paper, including proofs, is available on the web (White *et al.*, 1997).

2 Prediction Theories

PTs are expressed in the Temporal Calculus ($\mathcal{TC}$). $\mathcal{TC}$ is so called because it combines features of temporal logic with features analogous to those of the Situation Calculus. $\mathcal{TC}$ is a many sorted, higher order, temporal, nonmonotonic extension of Kleene's strong three-valued language (Kleene, 1952). A concise account of the relevant features of $\mathcal{TC}$ is now given.

Kleene introduced the third truth value *undefined* in order to be able to describe computations which may not, in principle, terminate. The language can also be given an epistemic, resource bounded, interpretation. Typically we have incomplete, or partial, information and our reasoning resources are limited. Consequently we cannot, as the classical ideal suggests, determine the truth or falsity of every sentence. Rather, we concentrate our efforts on attempting to establish the truth or falsity of those sentences that we consider relevant and ignore – treat as undefined – the truth values of the rest. Accordingly, a sentence can be true (established by the reasoner(s) as being true), false (established by the reasoner(s) as being false) or undefined (ignored by the reasoner(s) as irrelevant, or unestablishable within the reasoner(s) resource limitations). Appropriately, the truth conditions of the logical constants yield a Boolean truth value wherever possible. For example, $\neg\phi$ is true if ϕ is false, false if ϕ is true, and undefined otherwise. Similarly, $\phi \wedge \psi$ is true if both ϕ and ψ are both true, false if either is false, and undefined otherwise. Disjunction, $\vee$, and material implication, $\supset$, can be defined in the usual way. The language is then readily extended to the first-order case. Atomic sentences are interpreted using partial functions. Thus an atomic sentence, a *Kleene atom*, $r(u_1, \ldots, u_n)$ may be true, false or undefined in a model. A universally quantified sentence $\forall v\phi$ is then true if every instance of ϕ is true, false if one instance is false, and is undefined otherwise. The existential quantifier, $\exists$, is defined in the usual way.

As it stands, Kleene's logic is appropriate for representing monotonic reasoning with partial information. In order to be able to represent nonmonotonic reasoning, an *undefined* operator, ?, is added. Informally $?\phi$ states that the truth value of ϕ is undefined; that is, that neither the truth nor the falsity of ϕ is established. Thus $?\phi$ should be true if ϕ is undefined, and false otherwise. This operator can be used to define *weak affirmation*, *weak negation*, and *defined* operators, making it possible to express meta-level distinctions such as "ϕ is not false" and "ϕ is not true", and consequently nonmonotonic inference rules, in the object language:

$$\begin{array}{lcll} \circ\phi & \overset{\text{def}}{=} & ?\phi \vee \phi & \text{“}\phi\textit{ is not false”}, \\ \bullet\phi & \overset{\text{def}}{=} & ?\phi \vee \neg\phi & \text{“}\phi\textit{ is not true”}, \\ !\phi & \overset{\text{def}}{=} & \neg ?\phi & \text{“}\phi\textit{ is not undefined”}. \end{array}$$

Similar extensions to Kleene's language, starting with the weak negation operator, are discussed in (Doherty, 1996).

A drawback with Kleene's semantics is that the defined operator $\supset$ does not provide a satisfactory account of conditionality. In particular, if ϕ is true and ψ is undefined, then $\phi \supset \psi$ is undefined, rather than false. Moreover, in keeping with the resource-bounded interpretation, it is desirable to have a conditional which is true if the antecedent is not true, regardless of the truth value of the consequent. A conditional of this kind can be thought of as expressing a constraint which must be met if the antecedent is true, but which can otherwise be ignored. It can be defined as follows:

$$\phi \rightarrow \psi \quad \overset{\text{def}}{=} \quad \neg\bullet\phi \supset \neg\bullet\psi$$

where the sentence $\neg\bullet\phi$ makes the classical claim that ϕ is true; that is, it is true if ϕ is, and is false otherwise. The new conditional is itself classically valued. It is false if the antecedent is true and the consequent is not, and true otherwise. Kleene added his own classical operator. The sentence $\phi \equiv \psi$ states that ψ and ψ are equivalent; that is, that they always have the same truth value. The sentence is true if this is the case and is false otherwise. This operator can be defined as follows:

$$\phi \equiv \psi \quad \overset{\text{def}}{=} \quad (\neg\bullet\phi \wedge \neg\bullet\psi) \vee (\neg\circ\phi \wedge \neg\circ\psi) \vee (?\phi \wedge ?\psi)$$

where the sentence $\neg\circ\phi$ makes the classical claim that ϕ is false.

In order to represent time, an explicit temporal index is added to each atomic formula of the underlying language, to each Kleene atom; thus time is not reified, rather it is introduced as a separate sort and a predicative approach is adopted. For example the ordinary atom $r(u_1, \ldots, u_n)(t)$ states that the Kleene literal $r(u_1, \ldots, u_n)$ is true at the time point t. The temporal precedence operator, $<$, allows comparison of time points, thus $t < t'$ states that the time point t temporally precedes the time point t'. The semantics for time are as in standard temporal logic. Models now include a temporal frame $(\mathcal{T}, \mathcal{R}_\mathcal{T})$ where $\mathcal{R}_\mathcal{T} \subseteq \mathcal{T} \times \mathcal{T}$. Intuitively $\mathcal{T}$ is the set of time points and $(\mathcal{T}, \mathcal{R}_\mathcal{T})$ represents the flow of time. For convenience in this paper we will assume that time is discrete and linear. The truth value of Kleene literals may now vary from time point to time point.

As recommended by Galton (Galton, 1991) and as practised in the Situation Calculus, the difference in aspect between facts and events is signified. Accordingly, (primitive) events are introduced as a third sort.

In order to gain comparable expressivity with the Situation Calculus, *Kleene literals*, Kleene atoms and their negations, are added as a fourth sort. This higher-order quantification over literals is convenient, and can easily be replaced by treating Kleene literals as particulars and using a *Holds* predicate; as is done with fluents in the Situation Calculus. The final extension is a set of higher-order relations of the form $hr(e_1, \ldots, e_n, \lambda_1, \ldots, \lambda_m)(t)$; where $n + m > 0$, the e_i are event terms, and the λ_j are Kleene literals. For example, a sentence of the form $Occ(e)(t)$ states that a token of an event of type e occurs at t. Three further higher-order relations, *Aff*, *Qual* and *CAff*, will be introduced in the sequel.

The relevant formal semantics of $\mathcal{TC}$ are now outlined. A *model* of $\mathcal{TC}$ is a structure $\langle \mathcal{D}, \mathcal{E}, \mathcal{T}, \mathcal{R}_\mathcal{T}, \mathcal{F}, \mathcal{R}, \mathcal{HR}, \mathcal{V}\rangle$, where: $\mathcal{D}$ is the set of domain objects, $\mathcal{E}$ is the set of primitive events, $\langle \mathcal{T}, \mathcal{R}_\mathcal{T}\rangle$ is a (discrete, linear) temporal frame, $\mathcal{F}$ interprets the functions for each sort, $\mathcal{R}$ is a (possibly partial) function which interprets relations at each time point, $\mathcal{HR}$ is a (possibly partial) function which interprets higher-order relations at each time point, and $\mathcal{V}$ interprets the ground terms of each sort.

The interpretation of terms is straightforward. If M is a model with term interpretation function $\mathcal{V}$ and g is a variable assignment for M, then the term evaluation function $\mathcal{V}_g$ interprets ordinary, event, and temporal terms in the usual way, while literal variables are assigned literals, and literals are "interpreted" as themselves.

The truth and falsity conditions of sentences are then defined by means of the intermediary notions of satisfaction and violation.

Definition 2.1. *Let $M = \langle \mathcal{D}, \mathcal{E}, \mathcal{T}, \mathcal{R}_\mathcal{T}, \mathcal{F}, \mathcal{R}, \mathcal{HR}, \mathcal{V}\rangle$ be a $\mathcal{TC}$-model, g be a variable assignment for M, and ϕ be a $\mathcal{TC}$-formula. Then g* satisfies *ϕ in M (written $M, g \models \phi$) or* violates *ϕ in M (written $M, g =\!| \phi$) according to the clauses in Table 1.*

A formula ϕ is true *in a model M (written $M \models \phi$) if $M, g \models \phi$ for all variable assignments g. A formula ϕ is* false *in M (written $M =\!| \phi$) if $M, g =\!| \phi$ for all variable assignments g. We write $M \not\models \phi$ if it is not the case that $M \models \phi$, and $M \not=\!| \phi$ if it is not the case that $M =\!| \phi$. M is said to be a* model *of a set of sentences Θ (written $M \models \Theta$), if $M \models \phi$ for all $\phi \in \Theta$.*

A formula is said to be *monotonic* (at the level of evaluation (van Benthem, 1984)) if its satisfaction, or violation, is unaffected by an increase in information, otherwise the formula is said to be *nonmonotonic*. Formally, for models M and M' let $M \preceq M'$ mean that M' is at least as defined as M; that is, M and M' agree except perhaps that the functions $\mathcal{R}$ and $\mathcal{HR}$ of M may be less defined than the corresponding functions $\mathcal{R}'$ and $\mathcal{HR}'$ of M'; that is, $\mathcal{R}'$ agrees with $\mathcal{R}$ whenever $\mathcal{R}$ is defined, and similarly for $\mathcal{HR}'$ and $\mathcal{HR}$. Then a formula ϕ is monotonic if, for any assignment g and models M and M' such that $M \preceq M'$, if $M, g \models \phi$ then $M', g \models \phi$ and if $M, g =\!| \phi$ then $M', g =\!| \phi$. For example, any formula which does not contain the ? operator is monotonic,while a formula of the form $\bullet\lambda(t)$ is nonmonotonic; as, for given g and $M \prec M'$, it may be satisfied in M and violated in M'.

Prediction Theories (PTs) are expressed in $\mathcal{TC}$. Their definition is in part semantic as it refers to the earliest and latest time points at which rules are evaluated; see e.g. (Bell, 1996). For present purposes the details are not essential, so an informal outline is given. PTs may contain *boundary conditions*, *prediction rules*, and *domain rules.*

Boundary conditions are monotonic unconditional sentences which state facts and the occurrence of events; for example (5) below.

Domain rules are monotonic conditionals or biconditionals which are temporally undirected; for any instance of a domain rule, the evaluation of the antecedent and the consequent starts at the same time point and is completed at the same time point. Domain rules are used for a variety of purposes; for example, to state domain constraints, (7), and ramifications, (12).

Prediction rules are conditionals whose antecedents contain nonmonotonic formulas and which are temporally forwards-directed; for any instance of a prediction rule, the evaluation of the antecedent is completed at a time point which temporally precedes the earliest time point at which the evaluation of consequent is completed. Prediction rules are used to represent inertia and change.

In order to represent inertia, the *Aff* predicate is introduced. Intuitively, for Kleene literal λ and time point t, $Aff(\lambda)(t)$ states that the truth value of λ is affected at t; that is, that there is reason to doubt that the truth value of λ persists beyond t. A generic *inertia rule* can then be stated (as is customary, universal quantification is omitted):

$$\lambda(t) \wedge \bullet Aff(\lambda)(t) \rightarrow \lambda(t+1) \qquad (1)$$

Thus, if λ is true at t and it is not true that λ is affected at t, then λ remains true at $t+1$. The intention is to use this rule from left to right only in order to reason forwards in time from instances of its antecedent to instances of its consequent. Typically also we want to be able to infer the second conjunct of each instance nonmonotonically; whenever it is consistent to do so. Thus, on the intended interpretation of the inertia rule, Kleene literals, and consequently the formulas composed of them, persist by default.

In order to represent change, the *Qual* predicate is introduced. Intuitively, $Qual(e)(t)$ states that event e is qualified at time t; that is, that there is reason to doubt that e will succeed if e occurs at t. Then

$M,g \models t < t'$	iff	$(\mathcal{V}_g(t), \mathcal{V}_g(t')) \in \mathcal{R}_\mathcal{T}$
$M,g \dashv t < t'$	iff	$(\mathcal{V}_g(t), \mathcal{V}_g(t')) \notin \mathcal{R}_\mathcal{T}$
$M,g \models (u = u')(t)$	iff	$\mathcal{V}_g(u)$ is $\mathcal{V}_g(u')$
$M,g \dashv (u = u')(t)$	iff	$\mathcal{V}_g(u)$ is not $\mathcal{V}_g(u')$
$M,g \models r(u_1, \ldots, u_n)(t)$	iff	$\mathcal{R}(r, \mathcal{V}_g(t))(\mathcal{V}_g(u_1), \ldots, \mathcal{V}_g(u_n)) =$ true
$M,g \dashv r(u_1, \ldots, u_n)(t)$	iff	$\mathcal{R}(r, \mathcal{V}_g(t))(\mathcal{V}_g(u_1), \ldots, \mathcal{V}_g(u_n)) =$ false
$M,g \models hr(e_1, \ldots, e_n, \lambda_1, \ldots, \lambda_m)(t)$	iff	$\mathcal{HR}(hr, \mathcal{V}_g(t))(\mathcal{V}_g(e_1), \ldots, \mathcal{V}_g(e_n), \mathcal{V}_g(\lambda_1), \ldots, \mathcal{V}_g(\lambda_m)) =$ true
$M,g \dashv hr(e_1, \ldots, e_n, \lambda_1, \ldots, \lambda_m)(t)$	iff	$\mathcal{HR}(hr, \mathcal{V}_g(t))(\mathcal{V}_g(e_1), \ldots, \mathcal{V}_g(e_n), \mathcal{V}_g(\lambda_1), \ldots, \mathcal{V}_g(\lambda_m)) =$ false
$M,g \models \neg\psi$	iff	$M,g \dashv \psi$
$M,g \dashv \neg\psi$	iff	$M,g \models \psi$
$M,g \models ?\psi$	iff	neither $M,g \models \psi$ nor $M,g \dashv \psi$
$M,g \dashv ?\psi$	iff	either $M,g \models \psi$ or $M,g \dashv \psi$
$M,g \models \psi \land \chi$	iff	$M,g \models \psi$ and $M,g \models \chi$
$M,g \dashv \psi \land \chi$	iff	$M,g \dashv \psi$ or $M,g \dashv \chi$
$M,g \models \forall v \psi$	iff	$M,g' \models \psi$ for all g' such that $g \overset{v}{\approx} g'$
$M,g \dashv \forall v \psi$	iff	$M,g' \dashv \psi$ for some g' such that $g \overset{v}{\approx} g'$

Table 1: Semantics for the Temporal Calculus

a generic *change rule* can be stated as follows (again omitting the quantification):

$$Pre(e)(t) \land Occ(e)(t) \land \bullet Qual(e)(t) \rightarrow Post(e)(t+1) \quad (2)$$

Thus if event e occurs at t and the normal preconditions of e, $Pre(e)(t)$, are true at t and e is not qualified at t, then infer that the normal postconditions of e, $Post(e)(t)$, are true at $t+1$; that is, that e succeeds at t. Once again, the intention is to use this rule from left to right only in order to reason forwards in time from instances of its antecedent to instances of its consequent. Typically also we want to be able to infer the second conjunct of each instance nonmonotonically; whenever it is consistent to do so. Thus, on the intended interpretation of the change rule, events succeed by default if their normal preconditions are true. The success of an action can thus be defined as:

$$Succ(e)(t) \equiv Pre(e)(t) \land Occ(e)(t) \land \bullet Qual(e)(t) \quad (3)$$

In order to represent context-dependent ramifications, we introduce the binary relation *CAff* on events and literals. Intuitively $CAff(e, \lambda)(t)$ states that the occurrence of e conditionally affects λ at t. Rules can then be added stating the conditions under which this relation holds; thus in Example 2.6, rule (13) states that if an agent is holding an object and the agent moves, then the location of the object is conditionally affected. The example then illustrates how rules of this kind are used in combination with the following generic conditional affects rule:

$$\lambda(t) \land Succ(e)(t) \land CAff(e, \lambda)(t) \rightarrow Aff(\lambda)(t) \quad (4)$$

The intended interpretation of PTs is enforced by their pragmatics. The definition is based on the partiality of $\mathcal{TC}$. The relevant models are among those in which *information* (what is defined/established) is minimised chronologically. The effect is that prediction rules are interpreted from left to right only, and the appropriate nonmonotonic inferences are generated. However, simple chronological minimisation is insufficient as it does not resolve possible conflicts between (instances of) the inertia rule and the change rule. When there is such a conflict, it seems that the (instance of the) prediction rule should take precedence. This corresponds to the intuition that, other things being equal, if an event occurs and it can succeed (that is, if its normal preconditions are true), then it will succeed. In order to capture this principle, simple chronological minimisation is refined to ***prioritised chronological minimisation***. The minimisation is still chronological. However in order to give preference to potential change, qualifications (reasons why an event may not succeed as expected) are minimised in preference to affectations (reasons why a fact should not persist) at each time point. The following abbreviations are useful. Let M be a $\mathcal{TC}$ model with term interpretation function $\mathcal{V}$.

Then for any time point t:

$M_{\mathcal{R}}/t = \{\lambda(t') \mid \mathcal{V}(t') \leq t \text{ and } M \models !\lambda(t')\}$,
$M_{\mathcal{A}}/t = \{Aff(\lambda)(t') \mid \mathcal{V}(t') \leq t \text{ and } M \models !Aff(\lambda)(t')\}$,
$M_{\mathcal{O}}/t = \{Occ(e)(t') \mid \mathcal{V}(t') \leq t \text{ and } M \models !Occ(e)(t')\}$,
$M_{\mathcal{Q}}/t = \{Qual(e)(t') \mid \mathcal{V}(t') \leq t \text{ and } M \models !Qual(e)(t')\}$.

Thus $M_{\mathcal{R}}/t$ is the set of all Kleene literals which are defined (which are either true or false) in M up to t, and similarly $M_{\mathcal{A}}/t$, $M_{\mathcal{O}}/t$, and $M_{\mathcal{Q}}/t$ are respectively the sets of all *Aff*, *Occ* and *Qual* literals which are defined in M up to t.

Definition 2.2. *Let M and M' be $\mathcal{TC}$-models which differ at most on the interpretation of Kleene literals and the Occ, Aff and Qual predicates. Then M is* prioritised chronologically less defined *than M' (written $M \prec_{pcld} M'$) if there exists a time point t such that:*

- *$M_{\mathcal{R}}/t \subset M'_{\mathcal{R}}/t$ and $M_{\mathcal{O}}/t \subseteq M'_{\mathcal{O}}/t$, or*
- *$M_{\mathcal{R}}/t \subseteq M'_{\mathcal{R}}/t$ and $M_{\mathcal{O}}/t \subset M'_{\mathcal{O}}/t$, or*
- *$M_{\mathcal{R}}/t \subseteq M'_{\mathcal{R}}/t$, $M_{\mathcal{O}}/t \subseteq M'_{\mathcal{O}}/t$ and $M_{\mathcal{Q}}/t \subset M'_{\mathcal{Q}}/t$, or*
- *$M_{\mathcal{R}}/t \subseteq M'_{\mathcal{R}}/t$, $M_{\mathcal{O}}/t \subseteq M'_{\mathcal{O}}/t$, $M_{\mathcal{Q}}/t \subseteq M'_{\mathcal{Q}}/t$ and $M_{\mathcal{A}}/t \subset M'_{\mathcal{A}}/t$.*

Definition 2.3. *A model M is a* prioritised chronologically least defined *model (a* p.c.l.d. model*) of ϕ if $M \models \phi$ and there is no other model M' such that $M' \models \phi$ and $M' \prec_{pcld} M$. A PT Θ* predicts *a sentence ϕ, written $\Theta \approx_{pcld} \phi$, if ϕ is true in all p.c.l.d. models of Θ.*

Example 2.4. By way of illustration we give an example of a simple multi-agent scenario and then extend it. There are three locations, $L1$, $L2$ and $L3$ and two agents Stan, S, and Ollie, O. Initially, at time 0, Stan is at $L1$, $L2$ is clear and Ollie is at $L3$:

$$At(S, L1)(0) \wedge At(O, L3)(0) \qquad (5)$$
$$L1 \neq L2 \neq L3 \neq S \neq O \qquad (6)$$

The boundary conditions are stated in (5), while (6) is a unique names axiom (as names are temporally invariant, reference to time is omitted and $\neg(L1 = L2)(t)$ is abbreviated to $L1 \neq L2$, etc.). We also have the following domain constraint:

$$At(x, l1)(t) \wedge l1 \neq l2 \rightarrow \neg At(x, l2)(t) \qquad (7)$$

Rule (7) states a metaphysical ramification (that nothing can be in two distinct locations at the same time). Change occurs as a result of a successful move action. This has the following explicit precondition and postcondition:

$$Pre(Move(x, l1, l2))(t) \equiv At(x, l1)(t) \qquad (8)$$
$$Post(Move(x, l1, l2))(t) \equiv At(x, l2)(t) \qquad (9)$$

To begin with we show that everything works out well if at time 0 Stan moves to $L2$ and Ollie stays where he is. Let $\Theta = \{(1), (2), (5), \ldots, (9)\} \cup \{Occ(Move(S, L1, L2))(0)\}$. Then $\Theta \approx_{pcld} At(S, L2)(1) \wedge \neg At(S, L1)(1) \wedge At(O, L3)(1)$. As prioritised chronological minimisation gives preference to potential change, $?Qual(Move(x, l1, l2))(0)$ is true in all p.c.l.d. models of Θ, so the action succeeds as intended.

Example 2.5. In order to illustrate qualifications we assume that at time 0 Ollie also attempts to move to $L2$. We want to represent the fact that the two simultaneous move actions conflict, with the result that each cancels the effects that the other would have had had it occurred independently (Gelfond *et al.*, 1991). In order to do so we add the following rule:

$$Occ(Move(x, l1, l2)(t) \wedge x \neq y \rightarrow Qual(Occ(Move(y, l1, l2))(t) \qquad (10)$$

Thus a move action is qualified if another agent attempts to move to the same location simultaneously. Let $\Theta' = \Theta \cup \{(10), Occ(Move(O, L3, L2)(0)\}$. Then, as required, nothing changes: $\Theta' \approx_{pcld} \lambda(0) \equiv \lambda(1)$.

Example 2.6. In order to illustrate conditional, or context-dependent, ramifications we incorporate the ice-cream example (Baker, 1991). This time Stan is holding an ice cream when he moves from $L1$ to $L2$, and we should conclude that the ice cream is at $L2$ as a result. We thus add the following boundary conditions and ramification rule:

$$Holding(S, I)(0) \wedge At(I, L1)(0) \qquad (11)$$
$$Holding(x, y)(t) \wedge At(x, l)(t) \rightarrow \mathrm{At}(y, l)(t) \qquad (12)$$

However this is insufficient as there are unintended models of $\Theta \cup \{(11), (12)\}$ in which the ice cream remains at $L1$ after the move. These result from the unintended contrapositive reading of (12). In order to block this use of the rule we need to represent the fact that the move action changes the position of the ice cream *if* Stan is holding it when the move occurs. In order to do so we use the *CAff* predicate:

$$Holding(x, y)(t) \rightarrow CAff(Move(x, l1, l2), At(y, l1))(t) \wedge CAff(Move(x, l1, l2), \neg At(y, l2))(t) \qquad (13)$$

Thus (13) states that if an agent is holding an object when the agent moves then the position of the object is conditionally affected. Let $\Theta'' = \Theta \cup \{(3), (4), (11), (12), (13)\}$. Then, as required, $\Theta'' \approx_{pcld} At(S, L2)(1) \wedge At(I, L2)(1)$.

3 Model Building

We will be describing an algorithm for building models of these theories. We shall build them in temporal order: given a model "up to time t", we shall compute the $\prec_{pcld}$-minimal extensions of it to time $t+1$. The relevant sense of a model "up to time t" is given by

Definition 3.1. *Given a model M of a prediction theory, the time bounded model M/t is*

$$\begin{aligned} M/t \overset{\text{def}}{=} & \{\lambda(t') \in M \mid t' \leq t\} \cup \\ & \{Occ(e)(t') \in M \mid t' \leq t\} \cup \\ & \{Qual(e)(t') \in M \mid t' < t\} \cup \\ & \{Aff(\lambda)(t') \in M \mid t' < t\}. \end{aligned}$$

Our theories will be collections of rules of the form $B \cup D \cup \{(1), (2)\}$, where B is a set of boundary conditions and D is a set of domain rules; our only prediction rules are thus the generic rules (1) and (2).

The basic intuition behind the algorithm is as follows. Suppose, for the moment, we are in a situation with no qualified actions. Fix a model M/t and consider extensions of it to a a model $M/(t+1)$. There are a couple of fairly trivial ways in which might be able to make $M/(t+1)$ smaller (i.e. lower in the $\prec_{pcld}$-order). Suppose firstly that $M/(t+1)$ contains propositions of the form $Aff(\lambda)(t)$, but where $M/(t+1) - \{Aff(\lambda)(t)\}$ is still a model; then we can simply remove $Aff(\lambda)(t)$ and get a smaller model. Suppose, also, that there is a literal $\lambda(t+1)$ in $M/(t+1)$ which is not entailed by the union of the boundary conditions, the domain rules, and the literals carried across by inertia from M/t. We can then find a smaller model $M'/(t+1)$, essentially by carrying the same literals across by inertia, but removing $\lambda(t+1)$.

We now consider models in which we have applied the above two steps as much as we can (these are called *reduced* models). The dominant contribution to the "size" (i.e. the position in the $\prec_{pcld}$-order) of such a model will be the set $M_{\mathcal{A}}/t$; if we have two extensions M and M', and we know that $M_{\mathcal{A}}/t \subset M'_{\mathcal{A}}/t$, then automatically $M \prec_{pcld} M'$. So we should be trying to minimise $M_{\mathcal{A}}/t$. However, as $M_{\mathcal{A}}/t$ gets smaller, then – because our models satisfy (1) – the set of Kleene literals at $t+1$ which are carried over from M/t by inertia gets larger. So one approach to minimising $M_{\mathcal{A}}/t$ would be to try to find the largest consistent set of such literals, and then assign to it a suitable set $M_{\mathcal{A}}/t$ to get a model. This is basically what our algorithm does: we have to do some work to show that it does, in fact, produce minimal models, and that it finds all of them that there are to be found, but this is the basic idea of it.

Initially we will make several restrictions on the theories that we can handle, and then indicate how they can be removed. Firstly, we will confine ourselves to theories without any *CAff* rules or *Qual* predicates; and, secondly, we will confine ourselves to theories whose domain rules and boundary conditions are *essentially Horn.*

Definition 3.2. *A collection of domain rules and boundary conditions is essentially Horn if none of the boundary conditions is existentially quantified or disjunctive or nonmonotonic, and none of the consequents of the domain rules is existentially quantified or disjunctive or nonmonotonic.*

These restrictions can all be removed. *CAff* rules can be handled by quite small changes in the algorithm, although the proof of correctness becomes somewhat more complicated. The same remarks apply to *Qual* predicates. Finally, we can deal with theories that are not essentially Horn, again by changing the algorithm slightly, but at a grievous cost in its tractability.

3.1 The Example

Running Example 1. In order to illuminate the definitions, we will use a running example. The theory is given in Table 2. There are n blocks, $b_1, \ldots, b_n$, which are initially ($t = 0$) stacked up on location l_1; the agent a is initially at l_0. At time 0 the agent a picks up the block b_1; the only action postcondition is $\mathrm{Holding}(a, b_1)(1)$. The theory has $n+1$ minimal models at $t = 1$, given in Table 3. (These are, of course, not the full models, but enough propositions are given to specify the relevant model.) M_n is the intended model: the agent stays fixed, and all of the blocks move to where the agent is, i.e. l_0. M_0 is the model in which none of the blocks move, but the agent moves to where the blocks are; in the other models M_i exactly i blocks ($b_1, \ldots, b_i$) move to l_0, and we have $\neg\mathrm{On}(b_{i+1}, b_i)(1)$.

3.2 Preliminary Reductions

First some definitions. We will, in all this, be starting with a fixed model M/t, and we will be constructing extensions $M/(t+1)$ of it.

Domain Rules		
$\mathrm{At}(x,l)(t)$	$\rightarrow$	$\mathrm{loc}(l)$
$\mathrm{On}(x,l)(t) \wedge \mathrm{loc}(l) \wedge \mathrm{loc}(l') \wedge l \neq l'$	$\rightarrow$	$\neg\mathrm{On}(x,l')(t)$
$\mathrm{At}(x,l)(t) \wedge \mathrm{loc}(l) \wedge \mathrm{loc}(l') \wedge l \neq l'$	$\rightarrow$	$\neg\mathrm{At}(x,l')(t)$
$\mathrm{On}(x,b)(t) \wedge \mathrm{block}(b) \wedge \mathrm{block}(b') \wedge b \neq b'$	$\rightarrow$	$\neg\mathrm{On}(x,b')(t)$
$\mathrm{On}(x,b)(t) \wedge \mathrm{block}(b) \wedge \mathrm{loc}(l)$	$\rightarrow$	$\neg\mathrm{On}(x,l)(t)$
$\mathrm{On}(x,l)(t) \wedge \mathrm{block}(b) \wedge \mathrm{loc}(l)$	$\rightarrow$	$\neg\mathrm{On}(x,b)(t)$
$\mathrm{At}(a,l)(t) \wedge \mathrm{agent}(a) \wedge \mathrm{loc}(l)$	$\rightarrow$	$\mathrm{On}(a,l)(t)$
$\neg\mathrm{On}(a,l)(t) \wedge \mathrm{agent}(a) \wedge \mathrm{loc}(l)$	$\rightarrow$	$\neg\mathrm{At}(a,l)(t)$
$\mathrm{On}(a,l)(t) \wedge \mathrm{agent}(a) \wedge \mathrm{loc}(l)$	$\rightarrow$	$\mathrm{At}(a,l)(t)$
$\neg\mathrm{At}(a,l)(t) \wedge \mathrm{agent}(a) \wedge \mathrm{loc}(l)$	$\rightarrow$	$\neg\mathrm{On}(a,l)(t)$
$\mathrm{On}(b,l)(t) \wedge \mathrm{block}(b) \wedge \mathrm{loc}(l)$	$\rightarrow$	$\mathrm{At}(b,l)(t)$
$\neg\mathrm{At}(b,l)(t) \wedge \mathrm{block}(b) \wedge \mathrm{loc}(l)$	$\rightarrow$	$\neg\mathrm{On}(b,l)(t)$
$\mathrm{Holding}(a,x)(t) \wedge \mathrm{At}(a,l)(t)$	$\rightarrow$	$\mathrm{At}(x,l)(t)$
$\mathrm{Holding}(a,x)(t)$	$\rightarrow$	$\neg\mathrm{Handempty}(a)(t)$
$\mathrm{Handempty}(a)(t) \wedge \mathrm{block}(b)$	$\rightarrow$	$\neg\mathrm{Holding}(a,b)(t)$
Boundary Conditions		
$\mathrm{On}(b_1,l_1)(0) \wedge \mathrm{On}(b_2,b_1)(0) \wedge \ldots \wedge \mathrm{On}(b_n,b_{n-1})(0) \wedge$ $\mathrm{Handempty}(a)(0) \wedge \mathrm{At}(a,l_0)(0)$		
Action Postcondition		
$\mathrm{Holding}(a,b_1)(1)$		

Table 2: The Theory for The Running Example

M_0	$\mathrm{At}(a,l_1)$	$\mathrm{On}(b_2,b_1),\cdots,\mathrm{On}(b_n,b_{n-1})$	$\mathrm{At}(b_1,l_1),\cdots,\mathrm{At}(b_n,l_1)$
M_1	$\mathrm{At}(a,l_0)$	$\neg\mathrm{On}(b_2,b_1),\mathrm{On}(b_3,b_2),\cdots,\mathrm{On}(b_n,b_{n-1})$	$\mathrm{At}(b_1,l_0),\mathrm{At}(b_2,l_1),\cdots,\mathrm{At}(b_n,l_1)$
M_2	$\mathrm{At}(a,l_0)$	$\mathrm{On}(b_2,b_1),\neg\mathrm{On}(b_3,b_2),\cdots,\mathrm{On}(b_n,b_{n-1})$	$\cdots\mathrm{At}(b_2,l_0),\mathrm{At}(b_3,l_1),\cdots,\mathrm{At}(b_n,l_1)$
$\vdots$			
M_{n-1}	$\mathrm{At}(a,l_0)$	$\mathrm{On}(b_2,b_1),\cdots,\mathrm{On}(b_{n-2},b_{n-1}),\neg\mathrm{On}(b_n,b_{n-1})$	$\mathrm{At}(b_1,l_0),\cdots,\mathrm{At}(b_{n-1},l_0),\mathrm{At}(b_n,l_1)$
M_n	$\mathrm{At}(a,l_0)$	$\mathrm{On}(b_2,b_1),\cdots,\mathrm{On}(b_n,b_{n-1})$	$\mathrm{At}(b_1,l_0),\cdots,\mathrm{At}(b_n,l_0)$

Table 3: Models of the Theory at $t = 1$

$$\begin{aligned}
\mathcal{B} &= \emptyset,\\
\mathcal{P} &= \{\text{Holding}(a,b_1)(1)\},\\
\overline{\emptyset} &= \{\text{Holding}(a,b_1)(1),\\
&\quad \neg\text{Handempty}(a)(1)\},\\
\overline{\{\text{At}(a,l_0)(1)\}} &= \{\text{Holding}(a,b_1)(1),\\
&\quad \neg\text{Handempty}(a)(1), \text{At}(a,l_0)(1),\\
&\quad \text{On}(a,l_0)(1), \neg\text{At}(a,l_1)(1),\\
&\quad \neg\text{On}(a,l_1)(1), \text{At}(b_1,l_0)(1),\\
&\quad \neg\text{At}(b_1,l_1)(1), \neg\text{On}(b_1,l_1)(1)\}.
\end{aligned}$$

Table 4: The $\overline{(\cdot)}$ Operator in the Running Example

Definition 3.3. *Let $\mathcal{B}$ be the smallest set of literals which satisfy the boundary conditions at $t+1$. Let $\mathcal{P}$ be the postconditions of all actions e such that $M/t \models Succ(e)(t)$. For a set γ of literals with latest time point $t+1$, let $\overline{\gamma}$ be the smallest such set containing $\gamma \cup \mathcal{B} \cup \mathcal{P}$ and closed under the domain rules.*

Running Example 2. The effects of the $\overline{(\cdot)}$ operator in the example are shown in Table 4.

Definition 3.4. *For a given model $M/(t+1)$, let $\alpha_t(M) = \{\mathit{Aff}(\lambda)(t) \mid \lambda(t) \in M, \lambda(t+1) \notin M\}$, and let $M^\sharp/(t+1)$ be M with the subset $\{\mathit{Aff}(\lambda)(t) \mid \mathit{Aff}(\lambda)(t) \in M\}$ replaced by $\alpha_t(M)$.*

Proposition 3.5. *If $M/(t+1)$ is a model of the theory, then so is $M^\sharp$, and $M^\sharp \preceq_{pcld} M$.*

Running Example 3. We start with the set $\overline{\{\text{At}(a,l_0)(1)\}}$ of Running Example 2. Consider a model $M/1$ containing our given $M/0$ and whose literals at 1 are $\overline{\{\text{At}(a,l_0)(1)\}}$. Because all such models satisfy the generic inertia rule, any such $M/1$ must contain the following propositions:

$$\begin{aligned}
&\{\mathit{Aff}(\neg\text{Holding}(a,b_1))(0), \mathit{Aff}(\text{Handempty}(a))(0)\\
&\mathit{Aff}(\text{At}(b_1,l_1)(0), \mathit{Aff}(\neg\text{At}(b_1,l_0))(0),\\
&\mathit{Aff}(\text{On}(b_1,l_1))(0),\\
&\mathit{Aff}(\text{At}(b_2,l_1))(0), \mathit{Aff}(\neg\text{At}(b_2,l_0))(0)\\
&\mathit{Aff}(\text{On}(b_2,b_1))(0), \mathit{Aff}(\neg\text{On}(b_2,l_0))(0)\\
&\vdots\\
&\}
\end{aligned}$$

However, it may contain more: it may contain propositions such as $\mathit{Aff}(\text{At}(a,l_0))(0)$, which are not required by the generic inertia rule. For any model M, $\alpha_t(M)$ is the set of affected literals at t required by the generic inertia rule; the $\sharp$ operation replaces the set of affected literals in M by this minimal set.

Definition 3.6. *If M is a model, let $\iota_t(M) = \{\lambda(t+1) \in M \mid \lambda(t) \in M\}$, and let $M^\flat$ be M with $\{\lambda(t+1) \mid \lambda(t+1) \in M\}$ replaced by $\overline{\iota(M)}$.*

Proposition 3.7. *If M is a model, $M^\flat$ is also a model, and $M^\flat \preceq_{pcld} M$.*

Running Example 4. Consider, again, models containing $\overline{\text{At}(a,l_0)(1)}$. All of them must contain $\overline{\{\text{At}(a,l_0)(1)\}}$, but some of them may contain more; for example, we can add Holding$(a,b_2)(1)$ and, provided we add the rest of $\overline{\{\text{At}(a,l_0)(1), \text{Holding}(a,b_2)(1)\}}$, we will still have a model. However, Holding$(a,b_2)(1)$ is warranted neither by inertia nor by the action postconditions nor the domain rules. In our example, $\iota_0(M) = \overline{\{\text{At}(a,l_0)(1)\}}$.

If we want to search for minimal models, then, we can restrict our attention to models with $M = M^\sharp$ and $M = M^\flat$. We call such models *reduced* (or, strictly speaking, reduced at t, but t is fixed and often suppressed).

3.3 The Fundamental Correspondence

Lemma 3.8. *If M is a model, then $\alpha_t(M) = \{\mathit{Aff}(\lambda)(t) \mid \lambda(t) \in M, \lambda(t+1) \notin \iota_t(M)\}$.*

We can now establish the fundamental

Proposition 3.9. *If M and N are reduced models with $M/t = N/t$, then $M/(t+1) \preceq_{pcld} N/(t+1)$ iff $\iota(N) \subseteq \iota(M)$.*

So we are now reduced to searching over sets $\iota_t(M)$, for models M. We want to know when a set of literals at $t+1$ is an $\iota_t(M)$, for a reduced model $M/(t+1)$. We start by defining a suitable closure operation:

Proposition 3.10. *Let σ be a set of literals at $t+1$. Then the set*

$$\tau(\sigma) \quad = \quad \overline{\sigma} \cap \{\lambda(t+1) \mid \lambda(t) \in M/t\}$$

is the smallest $\iota_t(M)$ containing σ, for any $M/(t+1)$ extending M/t.

Corollary 3.11. *The sets $\iota_t(M)$ are those subsets of $\{\lambda(t+1) \mid \lambda(t) \in M/t\}$ which are consistent and fixed under the mapping $\sigma \mapsto \tau(\sigma)$.*

We thus have

Proposition 3.12. *The $\prec_{pcld}$-minimal extensions of M/t to $M/(t+1)$ correspond to subsets σ of $\{\lambda(t+1) \mid \lambda(t) \in M/t\}$ which are fixed by τ and such that, for any $\lambda(t) \in M/t$ with $\lambda(t+1) \notin \sigma$, $\overline{\sigma \cup \{\lambda(t+1)\}}$ is inconsistent.*

Running Example 5. Consider the model – N, say – corresponding to $\overline{\{\text{At}(a,l_0)(1)\}}$. N is far from $\prec_{pcld}$-minimal, since we have $N \prec_{pcld} M_1, N \prec_{pcld} M_2, \ldots,$

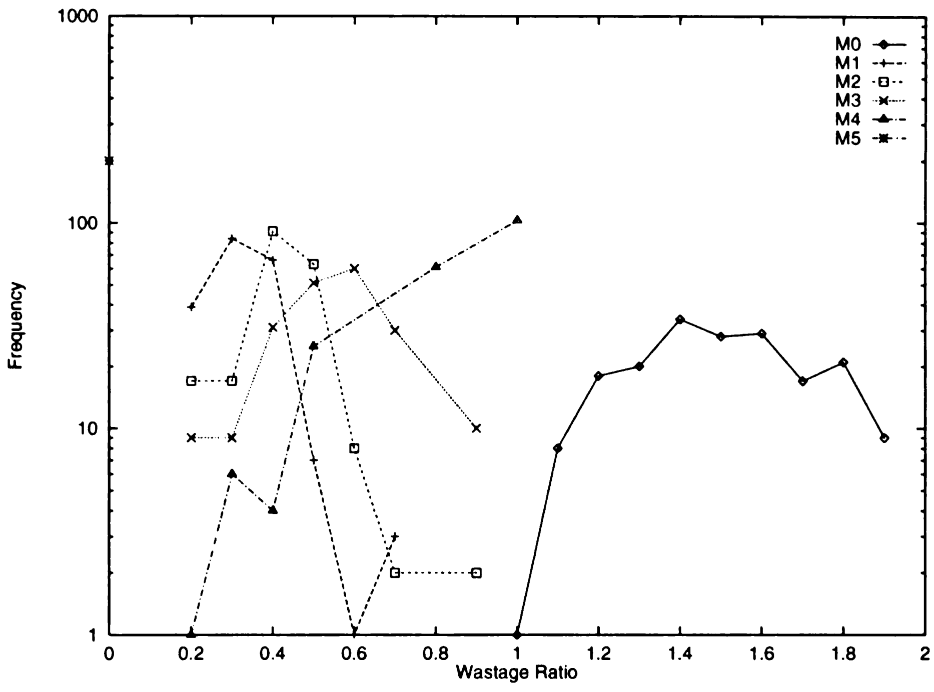


Figure 1: Number of Basic Augmentations for the Running Example

$N \prec_{pcld} M_n$. Correspondingly, $\iota(N) = \overline{\{\mathrm{At}(a,l_0)(1)\}}$ is small; by contrast, $\iota(M_1)$ is

$$\{\mathrm{At}(a,l_0)(1), \neg\mathrm{At}(a,l_1)(1), \mathrm{On}(a,l_0)(1), \neg\mathrm{On}(a,l_1)(1), \neg\mathrm{On}(b_1,b_2)(1), \neg\mathrm{On}(b_2,b_3)(1), \vdots \}$$

and, since we have $M_1 \prec_{pcld} N$, we have $\iota(N) \subsetneq \iota(M_1)$.

3.4 The Algorithm

We therefore want to find suitable subsets of $\{\lambda(t+1) \mid \lambda(t) \in M/t\}$. We first define

Definition 3.13. *Let $\sigma \subseteq \{\lambda(t+1) \mid \lambda(t) \in M/t\}$ be fixed under τ. A basic augmentation of σ is a set of the form $\tau(\sigma \cup \{\lambda(t+1)\})$ for some λ with $\lambda(t) \in M/t$, $\lambda(t+1) \notin \sigma$.*

An easy reformulation of the above results is:

Corollary 3.14. *If $\sigma \subseteq \{\lambda(t+1) \mid \lambda(t) \in M/t\}$ is fixed under τ, then it corresponds to a $\prec_{pcld}$-minimal model iff it has no basic augmentations.*

Running Example 6. Consider our model N of Running Example 5, corresponding to $\overline{\{\mathrm{At}(a,l_0)(1)\}}$. Basic augmentations will correspond to sets such as $\overline{\{\mathrm{At}(a,l_0)(1), \mathrm{On}(b_2,b_1)(1)\}}$, $\overline{\{\mathrm{At}(a,l_0)(1), \neg\mathrm{On}(b_2,b_3)(1)\}}$, and so on. Consider the model $\tilde{N}$ which corresponds to $\overline{\{\mathrm{At}(a,l_0)(1), \mathrm{On}(b_2,b_1)(1)\}}$.

$$\overline{\{\mathrm{At}(a,l_0)(1), \mathrm{On}(b_2,b_1)(1), \mathrm{On}(b_3,b_2)(1)\}}$$

corresponds to a basic augmentation of $\tilde{N}$, since it is still consistent, but

$$\overline{\{\mathrm{At}(a,l_0)(1), \mathrm{On}(b_2,b_1)(1), \mathrm{At}(b_3,l_1)(1)\}}$$

does not, because it is inconsistent.

The algorithm, then, is as follows.

Definition 3.15. *Given a model M/t and a theory, we start with $\tau(\emptyset)$ and, recursively, replace the set we have with one of its basic augmentations: when we find a set with no basic augmentations, we return that.*

Proposition 3.16. *The above algorithm terminates; it is correct – the sets it returns are of the form $\iota_t(M/(t+1))$, for $\prec_{pcld}$-minimal models $M/(t+1)$ – and it is complete – it will find any minimal extension using a suitable sequence of basic augmentations.*

3.5 Extensions

We aim to remove unwanted models by rules of the form $A \rightarrow \mathit{CAff}(e,\lambda)(t)$, for suitable antecedents A. Here $\mathit{CAff}(e,\lambda)(t)$ is true whenever $\mathit{Occ}(e)(t) \rightarrow \mathit{Aff}(\lambda)(t)$ is true. We can handle theories with CAff predicates in much the same way. The above algorithm has used, as a search space, the power set of

$\{\lambda(t+1) | \lambda(t) \in M/t\}$. If we have *CAff* predicates, we proceed as follows.

Definition 3.17. *Let $\mathcal{CA}$ be the set of literals*

$$\{\lambda(t+1) \quad | \quad \begin{array}{l} \textit{CAff}(e,\lambda)(t) \text{ is the conclusion of a} \\ \textit{CAff} \text{ rule with its premises} \\ \text{true in } M/t \text{ and where} \\ M \models \textit{Succ}(e)(t)\} \end{array}$$

Given this as the search space, we need to adjust the definitions of $\alpha_t(M)$ and $\iota_t(M)$ to

Definition 3.18. *For a given model M of a theory with CAff literals, let*

$$\begin{array}{rcl} \alpha_t(M) & = & \{\textit{Aff}(\lambda)(t) \mid \lambda(t) \in M, \lambda(t+1) \notin M\} \cup \\ & & \{\textit{Aff}(\lambda)(t) \mid \lambda(t+1) \in \mathcal{CA}\}, \\ \iota_t(M) & = & \{\lambda(t+1) \in M \mid \lambda(t) \in M\} - \mathcal{CA} \end{array}$$

With these modifications, everything works as before.

We can, by a similar redefinition, modify the algorithm to theories with a *Qual* predicate. We must define a new function on models, namely

Definition 3.19. *If $M/(t+1)$ is a model, then we define*

$$\begin{array}{rcl} \kappa_t(M/t+1) & = & \{\textit{Qual}(e)(t) \mid \\ & & M/t \models \textit{Pre}(e)(t) \wedge \textit{Occ}(e)(t), \\ & & M/(t+1) \not\models \textit{Post}(e)(t+1)\} \end{array}$$

We then define a new reduction operation on models, $M/(t+1) \mapsto M^{\natural}/(t+1)$, which replaces the set of actions qualified at t in M by $\kappa_t(M/(t+1))$. We prove again that $M^{\natural}$ is a model, and that $M^{\natural} \preceq_{pcld} M$.

We now show, as above, that we can represent the search space as a set of pairs $\langle\alpha, \iota\rangle$, where α is the set of actions which succeed at t and ι is the set $\iota_t(M)$. The ordering is lexicographic. We can now define an algorithm: we first search on α, finding a maximal set of actions the conjunction of whose postconditions is consistent together with the boundary conditions and domain rules. Having found such a maximal α, we carry out the usual search for a maximal ι as above, starting with the union of the postconditions of the actions in α. We can again prove correctness, termination, and completeness.

A rather more far-reaching modification is necessary if we have a theory which is not essentially Horn. In this case, we cannot use a closure operator for ensuring that the domain rules are satisfied. What we can do, however, is to use a suitable relation: we say $x\mathbf{R}y$, where x and y are sets of literals at $t+1$, whenever y contains x, satisfies the domain rules and boundary conditions, and is minimal among such subsets. We can define an algorithm using this relation. The major change lies in the definition of a basic augmentation:

Definition 3.20. *Let $\sigma \subseteq \{\lambda(t+1) \mid \lambda(t) \in M/t\}$ satisfy the domain rules and contain the boundary conditions at $t+1$ and the postconditions of all actions which succeed at t. A basic augmentation of σ is a set y such that $(\sigma \cup \{\lambda(t+1)\})\mathbf{R}y$ for some λ with $\lambda(t) \in M/t$, $\lambda(t+1) \notin \sigma$.*

With this redefinition, we again have an algorithm for which we can prove termination, completeness and correctness. However, it is much less tractable, because the relation $\mathbf{R}$ must be computed by a nondeterministic search.

3.6 Performance

3.6.1 Worst Case

Let us consider first a worst case analysis of the algorithm. We will evaluate this in terms of the size of the model at t:

Definition 3.21. *For a model M, $|M|_t$, the increase in the size of M at t, is*

$$|M/t| - |M/(t-1)|.$$

Furthermore, let K_t be the maximal size of a consistent set of literals at t, closed under the domain rules and the boundary conditions.

The execution of the algorithm is a series of basic augmentations, starting with $\tau(\emptyset)$; each basic augmentation must strictly increase the size of the model constructed. We terminate when there are no more basic augmentations to be found. Now $\tau(\emptyset)$ must be nonempty (otherwise the problem is essentially trivial; $M/(t-1)$ is simply copied over to M/t, and any sequence of basic augmentations will accomplish this). So the maximal number of basic augmentations is $(|M|_t) - 1$. Suppose that, after k basic augmentations, we have succeeded in finding a partial model N_k, and let $\tau(N_k) = \Sigma_k$. In searching for basic augmentations, we look at elements of $M/(t-1)$ which have not been carried over by inertia: there are at most $|M|_{t-1} - k$ of these. We might be unlucky, and have to look at all but one of these before we find one that gives a consistent extension: so we will have to investigate $|M|_{t-1} - k - 1$ of them before we add the appropriate one. So for each of these, we will have to show inconsistency: this may involve adding $K_t - |N_k|$ propositions. So at the worst case we may have to add $(|M|_{t-1} - k - 1)(K_t - |N_k|)$ propositions before we find the correct one to add. If α_k is the number of propositions added in the k'th basic augmentation itself, the total number of propositions added in the course of finding and constructing that augmentation comes out to be $(|M|_{t-1} - k - 1)(K_t - |N_k| + \alpha_k)$. Summing over k, we get

$$\Sigma_{k=0}^{k_1}(|M|_{t-1} - k - 1)(K_t - |N_k| + \alpha_k)$$

propositions in all, if there are, in all, k_1 basic augmentations. Now k_1 and the size of the α_i are related, because we have $|M|_t = \Sigma_{k=0}^{k_1}\alpha_k + |\tau(\emptyset)|$; since $(|M|_{t-1} - k - 1)$ is usually greater than 1, the above expression is largest when k_1 is largest - i.e. $|M|_t - |\tau(\emptyset)|$ - and all of the α_k are equal to 1. In this case, $|N_k| = \tau(\emptyset) + k$, and we are looking at the case where $\tau(\emptyset) = 1$. So the worst case complexity is, on this estimate,

$$\Sigma_{k=0}^{|M|_t - 1}(|M|_{t-1} - k - 1)(K_t - 1 - k + 1).$$

Now in practical cases, $|M|_{t-1}$, $|M|_t$ and K_t will all be bounded by some constant L, say, and the above estimate will be cubic in L.

3.7 Performance in Practice

In practice, however, both the average number of literals tried before a basic augmentation is found, and the average time needed to construct a basic augmentation, are small and bounded. We rarely have to try more than two literals before finding a basic augmentation, and quite often we find one immediately.

It is convenient to express this as follows. Suppose that we have construct a model M, and that $M_t - \tau(\emptyset) = K$, say; in the course of constructing the model, we will have successfully added K literals to it. Suppose also that, in the course of failed attempts to find basic augmentations, we add K' propositions; these propositions are, as it were, wasted. We define the *wastage ratio* to be K'/K. We find empirically that the wastage ratio is bounded, and, in fact, in all cases less than 2; this compares favourably with the worst case complexity analysis, for which the wastage ration would have increased quadratically with the size of the model.

This is shown in Figure 1. We applied the algorithm repeatedly to the scenario considered in the running example, with 5 blocks. The selection of literals for generating basic augmentations was randomised, and for each run we generated all of the models by reapplying the procedure to a consistent starting set of literals which was not contained in any of the models discovered so far. This biases the results somewhat, because certain models (and, in particular, the intended model) always tend to occur quite late in the sequence of models discovered, however we might try to randomise it. Because of this, the search procedure for these models starts off with quite a large initial set, since we have to make it quite large in order to make sure it is not contained in any of the models so far; and because of this unfair advantage, search for these models always seems to be unreasonably efficient (in fact, when we search for models late on in the sequence, most of the computational effort goes into finding a starting set which is consistent and not contained in any of the models so far, a task which can be exponential in the number of models already discovered). In practice we would cut down the number of models using *CAff* predicates, so this sort of behaviour would not arise.

Acknowledgements

This work forms part of the Dynamo Project and is supported by the United Kingdom Engineering and Physical Sciences Research Council under grant number GR/K19266.

References

Baker, A. 1991. Nonmonotonic Reasoning in the Framework of the Situation Calculus. *Artificial Intelligence*, **49**, 5–23.

Bell, J. 1991. Extended Causal Theories. *Artificial Intelligence*, **48**, 211–224.

Bell, J. 1995. Pragmatic Reasoning: A Model-Based Theory. *Pages 1–28 of:* Masuch, M., & Polos, L. (eds), *Applied Logic: How, What, and Why?* Amsterdam: Kluwer. A selection of papers from the Applied Logic Conference, Amsterdam, December 1992.

Bell, J. 1996. A Model-Based Approach to Predictive Causal Reasoning. *In:* (Doherty, 1996).

Bell, J. 1997. *Prediction Theories.* Working Paper.

Doherty, P. (ed). 1996. *Partiality, Modality, and Non-Monotonicity.* Stanford: CSLI Publications.

Galton, A. 1991. A Critique of Yoav Shoham's Theory of Causal Reasoning. *Pages 355–359 of: AAAI-91.* AAAI.

Gelfond, M., Lifschitz, V., & Rabinov, A. 1991. What are the limitations of the Situation Calculus? *Pages 167–169 of:* Boyer, R, (ed), *Essays in Honour of Woody Bledsoe.* Amsterdam: Kluwer.

Kleene, S. C. 1952. *Introduction to Metamathematics.* Amsterdam: North-Holland.

Shoham, Y. 1988. *Reasoning about Change.* Cambridge, MA: MIT Press.

van Benthem J. 1984. Partiality and Nonmonotonicity in Classical Logic. Center for the Study of Language and Information. Report CSLI-84-12. CSLI Publications, Stanford.

White, G., Bell, J., & Hodges, W. 1997. *Building Models of Prediction Theories.* Working paper. Available at `http://dcs.qmw.ac.uk/~graham/dynamo.html`

Probabilistic Reasoning

Reasoning About Infinite Random Structures with Relational Bayesian Networks

Manfred Jaeger
Max-Planck-Institut für Informatik
Im Stadtwald, 66123 Saarbrücken, Germany
jaeger@mpi-sb.mpg.de

Abstract

Relational Bayesian networks extend standard Bayesian networks by integrating some of the expressive power of first-order logic into the Bayesian network paradigm. As in the case of the related technique of knowledge based model construction, so far, decidable semantics only have been provided for finite stochastic domains. In this paper we extend the semantics of relational Bayesian networks, so that they also define probability distributions over countably infinite structures. Using a technique remeniscent of quantifier elimination methods in model theory, we show that probabilistic queries about these distributions are decidable.

1 INTRODUCTION

Bayesian networks (Pearl 1988) currently are the most popular and successful framework for representing and reasoning with probabilistic information. In their basic form, Bayesian networks define a probability distribution over the set of possible values of a finite set of random variables $X_1, \ldots, X_n$, each variable having a finite range of possible values.

Semantically, Bayesian networks describe *attributes* of individual random events or random objects, e.g. the symptoms and diseases of a random patient, or state variables describing a robot and its environment. For many applications the restriction to a fixed number of finite range random variables is a severe limitation. One of the approaches to go beyond these limits is the technique of *knowledge based model construction* (Wellman, Breese & Goldman 1992, Breese 1992, Haddawy 1994, Ngo & Haddawy 1995), where the number of variables is adjusted on a case-by-case basis. The basic idea here is to model the probabilistic domain not by a Bayesian network directly, but to use a knowledge base containing probabilistic rules that are used as a blueprint for the construction of Bayesian networks tailored to each specific inference task.

A simple example of a rule that such a knowledge base might contain is

$$\mathit{fever}(v) \overset{0.8}{\longleftarrow} \mathit{flu}(v), \tag{1}$$

with the intended meaning: if v suffers from the flu, then v will have fever with probability 0.8. A knowledge base containing such rules, and ground atoms *flu(thomas)*, *fever(sylvia)*, ..., gives rise to a standard Bayesian network over binary random variables representing all ground atoms relevant for the processing of a specific query.

When all the rules in the knowledge base only contain a single variable (as in (1)), then this rule-based approach only amounts to a notational variant of standard Bayesian networks, because information about one object mentioned in the knowledge base will not influence our inferences for another. Probabilistic knowledge bases gain their edge in expressive power over Bayesian networks by the ability to also define rules involving two or more variables, and n-ary relation symbols:

$$\mathit{infects}(u, v) \overset{0.6}{\longleftarrow} \mathit{sick}(u) \wedge \mathit{contact}(u, v). \tag{2}$$

A rule like (2) no longer merely describes the attributes of single random objects or events, but specifies *relations* between multiple random objects/events. A knowledge base with rules of this form, and ground atoms *infects(thomas,sylvia), sick(thomas),...* again can be used to generate a Bayesian network over ground atoms. Size and structure of this network now will depend on the number of constants appearing in the knowledge base. No single standard Bayesian network with a fixed number of finite-range random variables can be defined that supports the same inferences as can be drawn from a rule base with multi-variable rules.

In (Jaeger 1997) it was argued that the frameworks for knowledge based model construction proposed so far suffered from two deficiencies: first, they lack in expressiveness, second, the semantic clarity of the Bayesian network paradigm is lost, because the declarative character of the rules makes it hard to distill their meaning into a single probability distribution.

The issue of expressiveness was discussed at some length in (Jaeger 1997). The issue of semantic transparency it may be worthwhile to here elaborate on somewhat more. One of the main advantages of the Bayesian network paradigm is that it provides the user with a firm framework for how to describe a probability distribution: he or she is asked to specify the underlying directed acyclic graph, and the conditional probability tables for all nodes. Providing the required information then guarantees the user that one, and only one, probability distribution is being defined. Moreover, following this procedure gives the user a fairly well-understood control over the distribution he or she is defining. Probabilistic rules, like (1) or (2), on the other hand, only impose certain constraints on the probability distribution described. These constraints, hopefully, have a fairly intuitive meaning for the user, but will require a substantial overhead of semantic definitions and conventions in order to be given an exact interpretation. Breese (1992), for instance, gives the semantics of his knowledge bases indirectly by defining a procedure that, given a knowledge base and a specific query, constructs a Bayesian network in which the probability of the query is determined. Ngo and Haddawy (1995) provide declarative semantics for their representation formalism. However, a knowledge base in their language only defines a unique distribution if certain consistency and completeness conditions are satisfied. It is not clear that it is decidable whether these conditions are fulfilled, so that, in general, it may be impossible for the user to tell whether a knowledge base has semantics at all.

Relational Bayesian networks (RBNs) were proposed in (Jaeger 1997) as an alternative approach for the specification of probability distributions on relations between several random objects. Apart from providing additional expressiveness, RBNs recapture the semantic transparency afforded by the Bayesian network paradigm of probabilistic model construction (but see section 5 for a qualification).

In the present paper we are going to explore another advantage of RBNs: their semantics can be extended to define probability distributions for infinite domains of random objects, and queries for these distributions are still decidable. The framework of Ngo and Haddawy (1995) also is defined for infinite domains. Specifically, their distributions are defined for Herbrand universes as domains, which are infinite if the underlying first-order vocabulary contains at least one constant and one function symbol. Herbrand universes have a richer internal structure than the infinite domains for which the central results of the present paper are obtained. A result given in section 5 indicates why this structure on the domain might lead to inherent undecidability of Ngo and Haddawy's approach.

2 FINITE DOMAINS

2.1 THE BASIC FRAMEWORK

In this section we review the basic definitions introduced in (Jaeger 1997), extending them, where necessary, to deal with the case of infinite domains treated in the subsequent sections.

The purpose of an RBN is to define a probability distribution that models random attributes and random relations in a domain of objects or events. For the time being, assume that this domain $D = \{d_1, \ldots, d_n\}$ is finite. A probabilistic model for a set of relations $S = \{r_1, \ldots, r_k\}$ on this domain then consists of a probability distribution on the set of S-structures (a.k.a. models) with domain D, denoted $\text{Mod}_D(S)$. Relations in S can be of any arity. The arity of r_i is denoted $|r_i|$. For a single relation r_i, the set $\text{Mod}_D(r_i)$ of possible r_i-structures comprises just the set of possible interpretations $I(r_i) \subseteq D^{|r_i|}$ of r_i in D. Hence, $\text{Mod}_D(r_i)$ can be identified with the powerset of $D^{|r_i|}$. Moreover, an S-structure $\mathcal{M} \in \text{Mod}_D(S)$ is given by a tuple $(I(r_1), \ldots, I(r_k))$ of interpretations, so that $\text{Mod}_D(S)$ can be identified with the Cartesian product

$$\text{Mod}_D(S) = \text{Mod}_D(r_1) \times \ldots \times \text{Mod}_D(r_k). \quad (3)$$

Thus, a probability distribution on $\text{Mod}_D(S)$ can be defined in the form of a joint distribution for the individual $\text{Mod}_D(r_i)$. Viewing each $\text{Mod}_D(r_i)$ as a random variable, such a joint distribution can be defined following the Bayesian network paradigm: specify a directed acyclic graph with a node for each r_i, and at r_i define the conditional probability of each possible interpretation of r_i, given the instantiation of the parent nodes of r_i.

Assume that a directed acyclic graph has been given. For the node r_i denote by $Pa(r_i) = \{r_{j_1}, \ldots, r_{j_m}\}$ the set of parent nodes of r_i in the graph. Also, denote by $\mathcal{M}_i = (I(r_{j_1}), \ldots, I(r_{j_m}))$ some given $Pa(r_i)$-structure, which, in Bayesian network terminology, is just an instantiation of the parent nodes of r_i. For nodes without parents we interpret $\mathcal{M}_i$ simply as the given domain D. Following our program for defining a distribution on $\text{Mod}_D(S)$, we then have to define the conditional probability

$$P(I(r_i) \mid \mathcal{M}_i) \quad (4)$$

for every $I(r_i) \in \text{Mod}_D(r_i)$ and $\mathcal{M}_i \in \text{Mod}_D(Pa(r_i))$. Given some finite D, both $\text{Mod}_D(r_i)$ and $\text{Mod}_D(Pa(r_i))$

are finite, so that, in principle, all probabilities (4) could be explicitly listed in a huge conditional probability table. This, of course, is infeasible due to the size of such a table. More importantly, however, we aim for a generic definition that does not refer to a specific domain. To obtain such a generic definition, we define a schematic specification of conditional probabilities (4) that work "tuple by tuple", i.e. that determines for each $\boldsymbol{d} \in D^{|r_i|}$ the conditional probability

$$P(\boldsymbol{d} \in I(r_i) \mid \mathcal{M}_i). \tag{5}$$

Subsequently, we put

$$P(I(r_i) \mid \mathcal{M}_i) := \prod_{\boldsymbol{d} \in I(r_i)} P(\boldsymbol{d} \in I(r_i) \mid \mathcal{M}_i) \prod_{\boldsymbol{d} \notin I(r_i)} (1 - P(\boldsymbol{d} \in I(r_i) \mid \mathcal{M}_i)). \tag{6}$$

Conditional probabilities (5) we define by *probability formulas*, which are the key components of RBNs. To motivate the following definitions, consider a very simple example. Suppose we have the relation symbols $contact(u, v)$ and $sick(u)$, and want to model random structures in which the predicate *sick* depends on *contact*. Specifically, assume that for each person a, and for each $b \neq a$, $contact(a, b)$ is understood to be a possible cause for $sick(a)$ – each instance $contact(a, b)$ causing $sick(a)$ with probability 0.1, independently for different b. Then the overall probability of $sick(a)$, given complete information about *contact*, would be computed by combining probability values 0.1 by noisy-or for all instances $contact(a, b)$:

$$P(sick(a) \mid I(contact)) = n\text{-}o\{0.1 contact(a, b) \mid b \neq a\}.$$

The expression on the right hand side of this equation is an (informal) example of a probability formula. Key ingredients of probability formulas are *combination functions* (such as noisy-or) that are applied to multisets.

Definition 2.1 A *countable multiset* over $[0, 1]$ is a mapping $A : [0,1] \to \{0, 1, 2, \ldots\} \cup \{\omega\}$, where $A(q) > 0$ for at most countably many $q \in [0, 1]$. For countable multisets A, B we say that A is a subset of B ($A \subseteq B$), iff $A(q) \leq B(q)$ for all q. The supremum of a chain $(A_i)_{i\in\omega}$ ($A_i \subseteq A_j$ for $i \leq j$) of multisets is the multiset with $A(q) = sup\{A_i(q) \mid i \in \omega\}$. The multiset A is called finite, iff $A(q) \neq \omega$ for all q, and $A(q) > 0$ for only finitely many q.

For the present section only finite multisets are needed. We use the following notations to describe specific multisets: if $q_i \in [0, 1]$ for i from some countable index set I, then $\{q_i \mid i \in I\}$ denotes the multiset A with $A(q) = |\{i \mid q = q_i\}|$. Still more concretely, $\{q_1 : \lambda_1, \ldots, q_m : \lambda_m\}$ denotes the multiset A with $A(q_i) = \lambda_i$, $A(q) = 0$ for $q \notin \{q_1, \ldots, q_m\}$.

Definition 2.2 A *combination function* is any function that maps finite multiset over [0,1] into [0,1]. A combination function *comb* is *defined for countable multisets*, iff for every countable multiset A, and all chains $(A_i)_{i\in\omega}, (A'_i)_{i\in\omega}$ of finite multisets with $A = sup\{A_i \mid i \in \omega\} = sup\{A'_i \mid i \in \omega\}$, we have $sup\{comb(A_i) \mid i \in \omega\} = sup\{comb(A'_i) \mid i \in \omega\} =: comb(A)$.

Interesting examples of combination functions are

noisy-or:	$n\text{-}o\{a_i \mid i \in I\}$	$:= 1 - \prod_{i\in I}(1 - a_i)$
maximum:	$max\{a_i \mid i \in I\}$	$:= max\{a_i \mid i \in I\}$
mean:	$mean\{a_i \mid i \in I\}$	$:= \frac{1}{\lvert I\rvert} \sum_{i\in I} a_i$

It is easy to see that whenever *comb* is monotonically increasing (decreasing), i.e. $A \subseteq B$ implies $comb(A) \leq (\geq) comb(B)$, then *comb* is defined for countable multisets. Thus, noisy-or and max are defined for countable multisets. The mean, on the other hand, is not.

We need to introduce some notational conventions: logical variables (as opposed to random variables) are denoted throughout by letters u, v, w, z. Tuples $(v_1, \ldots, v_l)$ of variables are represented by a single letter $\boldsymbol{v}$ in bold face. The length of the tuple $\boldsymbol{v}$ is denoted $|\boldsymbol{v}|$. We also interpret $\boldsymbol{v}$ loosely as the set of variables it contains, so that expressions like $u \in \boldsymbol{v}$, or $\boldsymbol{w} \subseteq \boldsymbol{v}$ make sense. An *equality constraint* for $\boldsymbol{v}$ is a logical formula $c(\boldsymbol{v})$ that is a boolean combination of equality expressions $v_i = v_j$.

Definition 2.3 The class of *probability formulas* over the relational vocabulary S is inductively defined as follows.

(i) (Constants) Each rational number $q \in [0, 1]$ is a probability formula.

(ii) (Indicator functions) For every n-ary symbol $r \in S$, and every n-tuple $\boldsymbol{v}$ of variables, $r(\boldsymbol{v})$ is a probability formula.

(iii) (Convex combinations) When F_1, F_2, F_3 are probability formulas, then so is $F_1F_2 + (1 - F_1)F_3$.

(iv) (Combination functions) When $F_1, \ldots, F_k$ are probability formulas, *comb* is any combination function, $\boldsymbol{v}$,$\boldsymbol{w}$ are tuples of variables, and $c(\boldsymbol{v}, \boldsymbol{w})$ is an equality constraint, then $comb\{F_1, \ldots, F_k \mid \boldsymbol{w}; c(\boldsymbol{v}, \boldsymbol{w})\}$ is a probability formula.

A probability formula $F(\boldsymbol{v})$ over S maps $|\boldsymbol{v}|$- tuples $\boldsymbol{d}$ from the domain of a finite S-structure $\mathcal{M}$ into [0,1], according to the following definition.

Definition 2.4 Let $F(\boldsymbol{v})$ be a probability formula over S, D a finite domain, $\mathcal{M} \in \mathrm{Mod}_D(S)$, and $\boldsymbol{d} \in D^{|\boldsymbol{v}|}$. We inductively define the value $F(\boldsymbol{d})[\mathcal{M}]$.

(i) if $F = q$ then $F(\boldsymbol{d})[\mathcal{M}] = q$.

(ii) if $F = r(\boldsymbol{v})$ then $F(\boldsymbol{d})[\mathcal{M}] = 1$ if $\mathcal{M} \models r(\boldsymbol{d})$, and $F(\boldsymbol{d})[\mathcal{M}] = 0$ else.

(iii) if $F = F_1F_2 + (1 - F_1)F_3$ then $F(\boldsymbol{d})[\mathcal{M}] = F_1(\boldsymbol{d})[\mathcal{M}]F_2(\boldsymbol{d})[\mathcal{M}] + (1 - F_1(\boldsymbol{d})[\mathcal{M}])F_3(\boldsymbol{d})[\mathcal{M}]$.

(iv) if $F = comb\{F_1, \dots, F_k \mid \boldsymbol{w}; c(\boldsymbol{v}, \boldsymbol{w})\}$ then $F(\boldsymbol{d})[\mathcal{M}] = combA$, where A is the multiset that for $q \in [0, 1]$ has

$$A(q) = |\{(i, \boldsymbol{d}') \mid i \in \{1, \dots, k\}, \boldsymbol{d}' \in D^{|\boldsymbol{w}|}; \mathcal{M} \models c(\boldsymbol{d}, \boldsymbol{d}'), q = F_i(\boldsymbol{d}, \boldsymbol{d}')[\mathcal{M}]\}|.$$

If F only contains combination functions that are defined for countable multisets, then definition 2.4 also extends to countably infinite D. The following example illustrates case (iv) in the above definition.

Example 2.5 Let

$$F(v) = comb\{0.3r(v), 0.7s(v, w) \mid w; w \neq v\}. \quad (7)$$

Let $\mathcal{M}$ be an $\{r, s\}$-structure with domain $D = \{d_1, \dots, d_8\}$; let the interpretation of r in $\mathcal{M}$ be $\{d_1, \dots, d_4\}$, and let $\{(d_1, d_7), (d_1, d_8)\}$ be that part of the interpretation of s in $\mathcal{M}$ that has d_1 in the first component. To evaluate $F(d_1)[\mathcal{M}]$ we proceed as follows. First, we generate a list of all elements w of the domain that satisfy the constraint $w \neq d_1$. The result is $d_2, \dots, d_8$. For each tuple (d_1, d'), $d' \in \{d_2, \dots, d_8\}$, we compute $0.3r(v)[d_1, d'][\mathcal{M}]$ and $0.7s(v, w)[d_1, d'][\mathcal{M}]$. The notation $r(v)[d_1, d'], s(v, w)[d_1, d']$, rather than $r(d_1), s(d_1, d')$, here is used to emphasize that according to definition 2.4 we count each substitution of a tuple $(\boldsymbol{d}, \boldsymbol{d}')$ for the variables $(\boldsymbol{v}, \boldsymbol{w})$ in the F_i separately, no matter whether F_i actually contains the variables for which different values are substituted. The results of these recursive evaluations are 0.3 for the first formula, and 0 for the second, when $d' \in \{d_2, \dots, d_6\}$, and 0.3, respectively 0.7, for $d' \in \{d_7, d_8\}$. Here, the multiset A in (iv) thus is $\{0.3 : 7, 0 : 5, 0.7 : 2\}$. Applying *comb* to A then yields the result $F(d_1)[\mathcal{M}]$.

The following lemma contains a very useful result on the expressiveness of probability formulas. The simple proof can be found in (Jaeger 1997).

Lemma 2.6 Let $\phi(\boldsymbol{v})$ be a first-order formula in the relational vocabulary S. Then there exists a probability formula $F_\phi(\boldsymbol{v})$ in S, that uses *max* as the only combination function, s.t. for every finite S-structure $\mathcal{M}$, and every $\boldsymbol{d} \in D^{|\boldsymbol{v}|}$: $F_\phi(\boldsymbol{d}) = 1$ iff $\mathcal{M} \models \phi(\boldsymbol{d})$, and $F_\phi(\boldsymbol{d}) = 0$ else.

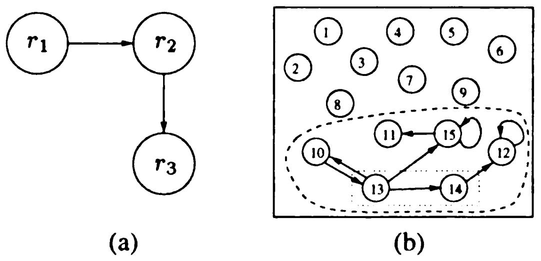


Figure 1: A relational Bayesian network and a typical structure

Definition 2.7 A *relational Bayesian network* for the (relational) vocabulary S is given by a directed acyclic graph containing one node for every $r \in S$. The node for an n-ary $r \in S$ is labeled with a probability formula $F_r(v_1, \dots, v_n)$ over the symbols in *Pa(r)*.

Given an RBN N and a finite domain D, we define the conditional probability $P_D^N(I(r_i) \mid \mathcal{M}_i)$ by substituting $F_{r_i}(\boldsymbol{d})[\mathcal{M}_i]$ for $P(\boldsymbol{d} \in I(r_i) \mid \mathcal{M}_i)$ in (6). This finally leads to the definition of the semantics of an RBN.

Definition 2.8 Let N be a relational Bayesian network over S, D a finite domain. N defines a probability measure P_D^N on $\mathrm{Mod}_D(S)$ by

$$P_D^N(I(r_1), \dots, I(r_k)) := \prod_{i=1}^{k} P_D^N(I(r_i) \mid \mathcal{M}_i). \quad (8)$$

We conclude this section with a very small example that we will refer to later on.

Example 2.9 Let $S = \{r_1, r_2, r_3\}$, where r_1 and r_3 are unary, r_2 is binary. An RBN N over S is defined via the graph in figure 1 (a), and the probability formulas

$$\begin{aligned} F_{r_1}(v) &\equiv 0.4 \\ F_{r_2}(v, w) &\equiv 0.2r_1(v)r_1(w) \\ F_{r_3}(v) &\equiv n\text{-}o\{0.3r_2(v, w) \mid w; w \neq v\} \end{aligned}$$

Figure 1 (b) shows one representative S-strucutre $\mathcal{M}$ over a domain D of 15 elements. The interpretation of r_1 in $\mathcal{M}$ is delimited in the figure by a dashed line, that of r_3 by a dotted line, and that of r_2 is represented by arrows. The probability of $\mathcal{M}$ is computed according to (8) by computing the three factors $P_D^N(I(r_i) \mid \mathcal{M}_i)$ $(i = 1, 2, 3)$, where $\mathcal{M}_3$ is the $\{r_2\}$-reduct of $\mathcal{M}$, $\mathcal{M}_2$ is the $\{r_1\}$-reduct, and $\mathcal{M}_1$ is just D. For each d_i $(i = 1, \dots, 15)$ in D we have $F_{r_1}(d_i)[\mathcal{M}_1] = 0.4$, so that according to (6), $P_D^N(I(r_1) \mid \mathcal{M}_1) = 0.4^6 \cdot 0.6^9$. The value of $F_{r_2}(d_i, d_j)[\mathcal{M}_2]$ is 0.2 when $i, j \in \{10, \dots, 15\}$, and 0 when at least one of i or j is not in $\{10, \dots, 15\}$. Thus, $P_D^N(I(r_2) \mid \mathcal{M}_2) = 0.2^6 \cdot 0.8^{30}$. The value of $F_{r_3}(d_i)[\mathcal{M}_3]$ is $1 - 0.7^{k_i}$, where k_i is the number of elements $d_j \neq d_i$ with $r_2(d_i, d_j)$. Thus, we get

$F_{r_3}(d_i)[\mathcal{M}_3] = 0$ for $i = 1, \ldots, 9, 11, 12$, $F_{r_3}(d_i)[\mathcal{M}_3] = 1-0.7^1$ for $i = 10, 14, 15$, and $F_{r_3}(d_i)[\mathcal{M}_3] = 1-0.7^3$ for $i = 13$, obtaining $P^N_D(I(r_3) \mid \mathcal{M}_3) = 0.3^2 \cdot 0.7^1 \cdot (1-0.7^3)$. Multiplying the three factors then yields $P^N_D(\mathcal{M})$.

2.2 RECURSIVE NETWORKS

In the distributions P^N_D defined via (6) and (8) strong independence assumptions hold: given the interpretation $\mathcal{M}_i$ of the parent relations of r_i, the events $r_i(\boldsymbol{d})$ and $r_i(\boldsymbol{d}')$ are independent for $\boldsymbol{d} \neq \boldsymbol{d}'$. As discussed in (Jaeger 1997), this is a serious limitation on what kind of probability distributions we are able to define with RBNs. Examples mentioned there of interesting types of relations that require a dependency of r-atoms are symmetric relations ($r(d, e)$ depends on $r(e, d)$), functional relations ($r(d, e)$ depends on $r(d, e')$ for all $e' \neq e$: exactly one of these atoms must be true), and temporal relations ($r(t, d)$ depends on $r(t-1, d)$).

On closer examination it turns out, though, that the assessment given in (Jaeger 1997) of the expressiveness of RBNs with respect to such dependencies is somewhat too pessimistic. While it is true that using an RBN with only one node r, we cannot define a distribution P^N_D on $\text{Mod}_D(r)$ with, e.g., $P^N_D(r(d, e)) = 1/2$, $P^N_D(r(d, e) \leftrightarrow r(e, d)) = 1$ for all d, e, we can define such a distribution by a network containing a second binary relation symbol s, and the two probability formulas

$$\begin{aligned} F_s(v, w) &:\equiv q \\ F_r(v, w) &:\equiv (1 - s(v, w))(1 - s(w, v)) \end{aligned}$$

where q is such that $(1 - q)^2 = 1/2$. In a similar manner, some forms of functional relations can be modeled with RBNs. Still, there are limits to what can be achieved within the framework presented so far. Temporal relations, for example, remain outside its scope.

To increase the expressiveness of RBNs, in (Jaeger 1997) *recursive RBNs* were defined. In a recursive RBN, probability formulas $F_r(\boldsymbol{v})$ are allowed to contain indicator functions of the form $r(\boldsymbol{w})$ in addition to the indicator functions $s(\boldsymbol{w})$ with $s \in Pa(r)$. The set of tuples $\boldsymbol{e}$ for which the evaluation of $F_r(\boldsymbol{d})$ requires the truth value of $r(\boldsymbol{e})$ now must be restricted in a way that makes the definition of $F_r(\boldsymbol{d})$ well-founded. For this purpose, we require that the domain D is equipped with suitable fixed relations or functions that enable us to define a well-founded partial order on $D^{|r|}$. Typical examples of such fixed relations and functions are a total order $<$ on D, or a successor function s. For the purpose of the present paper, we can limit ourselves to the case where D comes equipped with a built-in successor function (on finite domains D we take a successor function s to be undefined on the "last" element of D). The set

$$r\text{-}Pa(\boldsymbol{d}) := \{\boldsymbol{e} \mid F_r(\boldsymbol{d}) \text{ depends on } r(\boldsymbol{e})\}$$

now can be restricted by generalizing the equality constraints in probability formulas to constraints involving s. Here is an example of a probability formula that uses this generalization of the syntax to define a temporal kind of relation:

$$F_r(t, v) :\equiv$$
$$max\{\frac{1}{2}(1 - r(t', v)), r(t', v) \mid t'; s(t') = t\}. \quad (9)$$

For $(d, e) \in D^2$ here $r\text{-}Pa((d, e)) = \{(s^{-1}(d), e)\}$ ($= \emptyset$ if d is the "first" element of D). Given the truth value of $r(s^{-1}(d), e)$, the value of $F_r(d, e)$ is 1/2 if $r(s^{-1}(d), e)$ is false, and 1 if $r(s^{-1}(d), e)$ is true. We can think of r as a temporal property of e that becomes true at time d with probability 1/2 if not true already, and then remains true (because we have not introduced any machinery for dealing with multi-sorted domains, here the elements of D have to double as time points and as objects to which we ascribe property r).

As was shown in (Jaeger 1997), given a recursive probability formula $F_r(\boldsymbol{v})$ we can effectively compute a formula $r\text{-}pa(\boldsymbol{v}, \boldsymbol{w})$ over s, such that for all $\boldsymbol{d}, \boldsymbol{e} \in D^{|\boldsymbol{v}|}$:

$$\boldsymbol{e} \in r\text{-}Pa(\boldsymbol{d}) \text{ iff } (D, s) \models r\text{-}pa(\boldsymbol{d}, \boldsymbol{e}). \quad (10)$$

A recursive RBN now defines a probability distribution on $\text{Mod}_D(S)$ iff for all $r \in S$ the relation

$$\boldsymbol{e} \prec_r \boldsymbol{d} \;:\Leftrightarrow \boldsymbol{e} \in r\text{-}Pa(\boldsymbol{d}) \quad (11)$$

is well-founded. The resulting distribution P^N_D then still is explicitly defined by (6) and (8); only the terms $P(\boldsymbol{d} \in I(r_i) \mid \mathcal{M}_i)$ in (6) have to be replaced by

$$P(\boldsymbol{d} \in I(r_i) \mid \mathcal{M}_i, I(r_i\text{-}Pa(\boldsymbol{d}))),$$

where $I(r_i\text{-}Pa(\boldsymbol{d}))$ represents an interpretation of r_i restricted to $r_i\text{-}Pa(\boldsymbol{d})$.

Given a recursive network N, and a finite domain (D, s) it can be effectively decided whether the relations $\prec_r$ are well-founded, and if so, probabilities $P^N_D(r(\boldsymbol{d}))$ can be computed. Thus, the difference between recursive and non-recursive RBNs, for finite domains is of computational complexity, but not of a fundamental nature. In section 5 it will be shown that on infinite domains this changes drastically.

3 INFINITE DOMAINS: SEMANTICS

Even if the actual domains of random objects we encounter in the real world usually are finite, infinite domains are important for at least two reasons: they can afford conceptual simplicity, and they can be seen as the limiting case for large finite domains.

Conceptual simplicity is a reason for turning to infinite domains in cases where the actual (finite) domain is very large, and does not admit of a manageable description by a finite model. As an example, consider a model for a person's family tree. Even though the whole ancestry of that person, in fact, is finite, when we try to construct a formal model we will most likely end up with certain specifications – such as that everybody has two parents, and no one is ancestor of oneself – that only are satisfiable over an infinite domain of individuals. Hence, an infinite domain here would be a natural basis for, say, a probabilistic model of the propagation of genetic traits. In a similar vein, probabilistic models of language, for example as defined by stochastic context free grammars (see e.g. (Pynadath & Wellman 1996)), are defined on an idealized domain of infinitely many possible words and sentences, even though the collection of all sentences ever uttered is finite.

A somewhat different use for infinite domains is given when the existing finite domain admits of an adequate, manageable model, but the domain is large and difficult to determine exactly (cf. Bacchus et al. (1997)). An example of such an "open" domain is the set of all people that a given patient had contact with throughout his life. Here we may very well be able to specify suitable models for every domain D of finite size n by some generic description. Not knowing the appropriate n, however, rather than experimenting with arbitrarily chosen numbers, we may regard the limiting case of an infinite domain as the canonical approximation to the unknown domain. This, of course, only makes sense when we can apply the model we have designed for finite domains to an infinite domain as well, and when the properties of the model in the infinite case reflect the limiting behavior of large finite models.

Thus, there is an essential difference between the use of infinite models as conceptual idealizations, and as limiting cases: in the first use it is not expected that the infinite model in any way reflects properties of specific finite models. Inferences we draw in the infinite model we can accept without further ado for what they are: statements that are true in the infinite structure that we study for its own sake. In the second use, on the other hand, results inferred from the infinite model only are relevant in conjunction with the knowledge that (approximately) the same results will hold in all sufficiently large finite models.

In this and the following sections we show how to extend the semantics and inference techniques for RBNs to infinite domains. This makes RBNs available as probabilistic models for infinite structures in their first use. In a companion paper (Jaeger 1998) the problem is studied of when these infinite models can be seen as the limiting case of finite models, and a certain subclass of RBNs is identified that define infinite models suitable for employment in the second type of application.

Given an RBN N, and a countably infinite domain D, we need to define the probability distribution P_D^N induced by N on $\mathrm{Mod}_D(S)$. This definition will essentially follow the same pattern as in the case of finite domains. The main task is to substitute suitable continuous concepts where the discrete ones used in the finite case no longer work. In the following, only an overview of the construction of P_D^N is given, leaving out several standard measure theoretic arguments needed to fully justify the construction. It should be pointed out, however, that some of these arguments crucially depend on the countability of D, so that our construction does not carry over to uncountable D.

For simplicity, it is assumed throughout that $D = \omega = \{1, 2, \ldots\}$, and that $Pa(r_i) \subseteq S_i := \{r_1, \ldots, r_{i-1}\}$ holds in N for all i. When interpreted over $D = \omega$, each node r_i in N has the set of possible values $\mathrm{Mod}_\omega(r_i)$, which can be identified with the powerset $\omega^{|r_i|}$, and hence is uncountable. Specifically, the whole space $\mathrm{Mod}_\omega(S)$ is uncountable, and therefore needs to be equipped with a σ-algebra of measurable subsets before probability measures on that space can be defined. For our purpose a suitable σ-algebra $\mathfrak{A}_S$ is canonically defined as follows.

For $1 \leq i \leq k$ let $\mathfrak{E}_i$ contain all subsets $E \subseteq \mathrm{Mod}_\omega(r_i)$ of the form

$$\begin{aligned} E = E((\boldsymbol{d}_j)_j, (\boldsymbol{d}'_h)_h) &:= \\ \{\mathcal{M} \in \mathrm{Mod}_\omega(r_i) \mid \mathcal{M} &\models r_i(\boldsymbol{d}_j), \mathcal{M} \models \neg r_i(\boldsymbol{d}'_h); \\ & j = 1, \ldots, l, h = 1, \ldots, m\}, \end{aligned}$$

where $l, m \in \omega$, and $\boldsymbol{d}_j, \boldsymbol{d}'_h \in \omega^{|r_i|}$. Let $\mathfrak{A}_{r_i}$ be the σ-algebra generated by $\mathfrak{E}_i$. Define $\mathfrak{A}_S$ to be the product σ-algebra $\mathfrak{A}_{r_1} \otimes \ldots \otimes \mathfrak{A}_{r_k}$.

It is no longer possible to define conditional probabilities $P(I(r_i) \mid \mathcal{M}_i)$ "pointwise" for each $I(r_i) \in \mathrm{Mod}_\omega(r_i)$ and $\mathcal{M}_i \in \mathrm{Mod}_\omega(Pa(r_i))$, because any specific $I(r_i)$ or $\mathcal{M}_i$ typically will have probability 0. Instead, we have to replace the definition of $P(I(r_i) \mid \mathcal{M}_i)$ in (6) by a definition of *transition probabilities* $K_i(\mathcal{M}_i, A)$ from $\mathcal{M}_i$ to measurable subsets $A \in \mathfrak{A}_{r_i}$. Functions $K(\cdot, \cdot)$ that represent such transition probabilities are known as *stochastic kernels* (see e.g. (Jacobs 1978)).

It is sufficient to define the values $K_i(\mathcal{M}_i, A)$ for $A = E \in \mathfrak{E}_i$. In analogy to (6), for $E = E((\boldsymbol{d}_j)_j, (\boldsymbol{d}'_h)_h) \in \mathfrak{E}_i$, we would like to define

$$K_i(\mathcal{M}_i, E) := \prod_{j=1}^{l} F_{r_i}(\boldsymbol{d}_j)[\mathcal{M}_i] \prod_{h=1}^{m} (1 - F_{r_i}(\boldsymbol{d}'_h)[\mathcal{M}_i]). \quad (12)$$

It can be shown that (12) defines a stochastic kernel from $\mathrm{Mod}_\omega(Pa(r_i))$ to $\mathrm{Mod}_\omega(r_i)$, provided that F_{r_i} only contains combination functions satisfying a certain measurability condition. We do not go into the details here, and only note that virtually every combination function of practical interest will satisfy this condition. We call a probability formula *admissible*, if it only contains combination functions satisfying the measurability condition. Similarly, an RBN is called admissible, if it only contains admissible probability formulas.

When via (12) stochastic kernels $K_1, \ldots, K_k$ have been defined, in analogy to (8), a probability measure P_ω^N on $\mathfrak{A}_S$ is defined by

$$P_\omega^N(A) = \int\int\cdots\int 1_A(I_1,\ldots,I_k)K_k((I_1,\ldots,I_{k-1}),dI_k)\ldots \ldots K_2(I_1,dI_2)K_1(dI_1), \qquad (13)$$

where $A \in \mathfrak{A}_S$, 1_A is the indicator function [1] of A, and the I_i are integration variables ranging over $\mathrm{Mod}_\omega(r_i)$.

4 INFINITE DOMAINS: INFERENCE

In this section the main technical results are derived: it is shown that queries for the probability distributions P_ω^N are decidable.

We first give an outline of the argument by an informal derivation of the probability $P_\omega^N(r_3(d))$ for the network N from example 2.9, and an arbitrary $d \in \omega$. We try to determine $P_\omega^N(r_3(d))$ by reasoning about the form of structures $\mathcal{M} \in \mathrm{Mod}_\omega(S)$ that are "typical" according to P_ω^N. First, we consider the expected interpretation of r_1: since each $d \in \omega$ has a positive probability both of belonging to r_1, and of not belonging to r_1, and since membership in r_1 is determined independently for distinct d, we know by probabilistic 0-1 laws that with probability 1 $\mathcal{M}$ will contain infinitely many elements both within and outside r_1.

Next, we consider the expected interpretation of r_2. For each given element d with $r_1(d)$, and each d' with $r_1(d')$ there is a positive probability that $r_2(d,d')$ holds. Since this is independent for different d', and because, according to our first result, there are infinitely many candidates d', by the same 0-1 laws as above, we know that with probability 1 there will exist infinitely many d' with $r_2(d,d')$. Since there are only countably many different d, this even means that with probability 1 for every d with $r_1(d)$ there will exist infinitely many d' with $r_2(d,d')$. For d with $\neg r_1(d)$, on the other hand, with probability 1 there exist no d' with $r_2(d,d')$. Thus, in a typical structure $\mathcal{M}$ we have $F_{r_3}(d)[\mathcal{M}] = 0$ for d with $\neg r_1(d)$, and $F_{r_3}(d)[\mathcal{M}] = 1$ for d with $r_1(d)$. Hence, for an arbitrary $d \in \omega$: $P_\omega^N(r_3(d)) = P_\omega^N(r_1(d)) = 0.4$.

[1] In the usual measure-theoretic sense, not as appearing in definition 2.3.

4.1 ALMOST SURE PROPERTIES

In the above derivation we have reasoned that with probability 1 structures $\mathcal{M} \in \mathrm{Mod}_\omega(S)$ will possess certain structural properties. Our first step in developing a general and rigorous method for computing probabilities in P_ω^N will be to provide a well-defined class of such structural properties in terms of syntactic characterizations. This is done in definition 4.1 We then, in theorem 4.5, show that each of these structural properties is either satisfied by almost every structure in $\mathrm{Mod}_\omega(S)$, or by almost none. Theorem 4.5 allows us to ignore for the computation of probabilities in P_ω^N all those structures that do not possess the canonical structural properties. This, in conjunction with the fact that in canonical structures $\mathcal{M}$ the functions $F_{r_i}(\cdot)[\mathcal{M}]$ have a very regular behavior (lemma 4.2), allows us to develop an effective decision procedure for P_ω^N in section 4.2.

The following definition introduces (variants of) concepts that are commonly used in finite model theory: types and extension axioms. Intuitively, an S- type $\tau(v_1,\ldots,v_k)$ is a formula that gives an explicit, complete description of an S-structure of size k.

Definition 4.1 Let S be a relational vocabulary. An *S-type* in the variables $\boldsymbol{v} = v_0,\ldots,v_k$ is a maximal consistent conjunction $\tau(\boldsymbol{v})$ of atomic and negated atomic formulas over S in the variables $\boldsymbol{v}$. A type $\tau(\boldsymbol{v})$ is called *proper* if it contains all the formulas $v_i \neq v_j$ ($v_i, v_j \in \boldsymbol{v}, i \neq j$). A type $\sigma(\boldsymbol{v}, w)$ *extends* the type $\tau(\boldsymbol{v})$, written $\tau \subset \sigma$, if every conjunct of τ is a conjunct in σ.

Let $\tau \subset \sigma$ be proper types. The sentence

$$\forall \boldsymbol{v}(\tau(\boldsymbol{v}) \rightarrow \neg\exists w\sigma(\boldsymbol{v}, w)). \qquad (14)$$

is called a *no-extension axiom* for τ, σ. The set of all sentences

$$\forall \boldsymbol{v}(\tau(\boldsymbol{v}) \rightarrow \exists^{\geq n} w\sigma(\boldsymbol{v}, w)) \quad (n \in \omega) \qquad (15)$$

we call an *ω-extension axiom*, and denote it by the (non first-order) formula $\forall \boldsymbol{v}(\tau(\boldsymbol{v}) \rightarrow \exists^\omega w\sigma(\boldsymbol{v}, w))$ (the quantifier $\exists^\omega$ thus is to be read as "there exist infinitely many"). An *extension theory* is a consistent set Φ_{Ext} of no-extension and ω-extension axioms, s.t. for every pair of proper types $\tau(\boldsymbol{v}) \subset \sigma(\boldsymbol{v}, w)$, Φ_{Ext} contains (14) or (15).

The no-extension axiom (14) is in fact logically equivalent to the simpler sentence $\neg\exists \boldsymbol{v} w\sigma(\boldsymbol{v}, w)$. To enable a more uniform treatment of the two types of axioms, we here use the syntactically more complicated form (14).

Lemma 4.2 Let $F(v_1,\ldots,v_n)$ be an admissible probability formula over S. Let Φ_{Ext} be an extension theory for S. Let $\mathcal{M} \in \mathrm{Mod}_\omega(S)$ with $\mathcal{M} \models \Phi_{\mathrm{Ext}}$. Let $\boldsymbol{d}, \boldsymbol{d'} \in \omega^{|n|}$ so that $\mathcal{M} \models \tau(\boldsymbol{d}) \wedge \tau(\boldsymbol{d'})$ for some type τ. Then $F(\boldsymbol{d})[\mathcal{M}] = F(\boldsymbol{d'})[\mathcal{M}]$.

The lemma is proved using a standard model-theoretic back-and-forth argument, which shows that under the given assumptions there is an automorphism of $\mathcal{M}$ that maps $\boldsymbol{d}$ to $\boldsymbol{d}'$. This means that $\boldsymbol{d}$ and $\boldsymbol{d}'$ satisfy the same first-order formulas in $\mathcal{M}$. The first-order theory of $\boldsymbol{d}$ in $\mathcal{M}$ uniquely determines the value $F(\boldsymbol{d})[\mathcal{M}]$.

The following definition and lemma are merely technical tools that allow us to take somewhat more information about extensions out of an extension theory, than we explicitly put in.

Definition 4.3 Let $\boldsymbol{w} = (w_1, \ldots, w_l)$. Let $\tau(\boldsymbol{v}) \subset \sigma(\boldsymbol{v}, \boldsymbol{w})$ be types. A *generalized extension axiom* is a formula of the form

$$\forall \boldsymbol{v}(\tau(\boldsymbol{v}) \rightarrow Q\boldsymbol{w}\sigma(\boldsymbol{v}, \boldsymbol{w}))$$

where $Q \in \{\neg\exists, \exists^{=1}, \exists^{\omega}\}$. The meaning of $\exists^{=1}$ is to be read as "there exists exactly one tuple", that of $\exists^{\omega}$ as "there exist infinitely many tuples". Specifically, $\exists^{\omega} w_1 w_2 \sigma(\boldsymbol{v}, w_1, w_2)$ already is true when $\exists^{=1} w_1 \exists^{\omega} w_2 \sigma(\boldsymbol{v}, w_1, w_2)$ holds.

For the formulation of the lemma, and also for subsequent use, we introduce the notation $\boldsymbol{w} \sqsubseteq \boldsymbol{v}$ as shorthand for the formula $\bigwedge_{w \in \boldsymbol{w}} \bigvee_{v \in \boldsymbol{v}} w = v$.

Lemma 4.4 Let Φ_{Ext} be an S-extension theory. Let $\tau(\boldsymbol{v}) \subset \sigma(\boldsymbol{v}, \boldsymbol{w})$ be S-types. Then

$$\Phi_{\text{Ext}} \models \forall \boldsymbol{v}(\tau(\boldsymbol{v}) \rightarrow Q\boldsymbol{w}\sigma(\boldsymbol{v}, \boldsymbol{w})), \qquad (16)$$

for either $Q = \neg\exists$, $Q = \exists^{=1}$, or $Q = \exists^{\omega}$. The case $Q = \exists^{=1}$ holds exactly when $\sigma(\boldsymbol{v}, \boldsymbol{w}) \models \boldsymbol{w} \sqsubseteq \boldsymbol{v}$.

The proof of the lemma is fairly straightforward by reducing the generalized extension axiom to several no-extension and ω-extension axioms in the sense of definition 4.1. The following is the main theorem in this section.

Theorem 4.5 Let N be an admissible relational Bayesian network over S. For each $i = 1, \ldots, k$ then there exists an extension theory $\Phi^N_{\text{Ext}}(S_i)$ for S_i with $P^N_{\omega}(\Phi^N_{\text{Ext}}(S_i)) = 1$.

Proof: We proceed by induction on $i = 1, \ldots, k$. The base case $i = 1$ is a simpler variation of the induction step, and is here omitted. Thus, let $i > 1$, and assume that $P^N_{\omega}(\Phi^N_{\text{Ext}}(S_{i-1})) = 1$ for some extension theory $\Phi^N_{\text{Ext}}(S_{i-1})$. In the following, for an S_{i-1}-structure $\mathcal{M}$, and an S_i-theory Φ, we write $K(\mathcal{M}, \Phi)$ for $K(\mathcal{M}, R)$, where $R := \{I(r_i) \subseteq \omega^{|r_i|} \mid (\mathcal{M}, I(r_i)) \models \Phi\}$.

To prove the theorem for i, we show that there exists an extension theory $\Phi^N_{\text{Ext}}(S_i)$, such that for every $\mathcal{M} \models \Phi^N_{\text{Ext}}(S_{i-1})$

$$K(\mathcal{M}, \Phi^N_{\text{Ext}}(S_i)) = 1. \qquad (17)$$

To prove (17) it is sufficient to show that for every pair $\tau(\boldsymbol{v}) \subset \sigma(\boldsymbol{v}, w)$ of proper S_i-types we have

$$K(\mathcal{M}, \forall \boldsymbol{v}(\tau(\boldsymbol{v}) \rightarrow Qw\sigma(\boldsymbol{v}, w))) = 1 \qquad (18)$$

for $Q = \neg\exists$, or $Q = \exists^{\omega}$. Since there are only countably many such pairs τ, σ, from (18) we obtain (17) by letting $\Phi^N_{\text{Ext}}(S_i)$ be the collection of all extension axioms for which (18) holds.

To show (18), we first partition $\tau(\boldsymbol{v})$ and $\sigma(\boldsymbol{v}, w)$ into two, respectively four, conjuncts:

$$\begin{aligned} \tau(\boldsymbol{v}) &\equiv \tau^{S_{i-1}}(\boldsymbol{v}) \wedge \tau^{r_i}(\boldsymbol{v}) \\ \sigma(\boldsymbol{v}, w) &\equiv \tau^{S_{i-1}}(\boldsymbol{v}) \wedge \tau^{r_i}(\boldsymbol{v}) \\ &\quad \wedge \sigma^{S_{i-1}, w}(\boldsymbol{v}, w) \wedge \sigma^{r_i, w}(\boldsymbol{v}, w) \end{aligned}$$

where $\tau^{S_{i-1}}(\boldsymbol{v})$ contains all conjuncts of $\tau(\boldsymbol{v})$ that are S_{i-1}-literals (including the (in-)equality formulas), $\tau^{r_i}(\boldsymbol{v})$ contains all r_i-literals of $\tau(\boldsymbol{v})$, $\sigma^{S_{i-1}, w}(\boldsymbol{v}, w)$ contains all S_{i-1}-literals of $\sigma(\boldsymbol{v}, w)$ that contain the variable w, and $\sigma^{r_i, w}(\boldsymbol{v}, w)$ contains all r_i-literals of $\sigma(\boldsymbol{v}, w)$ containing w.

We now consider the slightly strengthened axiom

$$\begin{aligned} \chi(Q) :\equiv \forall \boldsymbol{v}(\tau^{S_{i-1}}(\boldsymbol{v}) \rightarrow \\ Qw(\sigma^{S_{i-1}, w}(\boldsymbol{v}, w) \wedge \sigma^{r_i, w}(\boldsymbol{v}, w))) \end{aligned} \qquad (19)$$

($Q \in \{\neg\exists, \exists^{\omega}\}$), which is equivalent to the finite collection of extension axioms obtained by replacing $\tau^{S_{i-1}}(\boldsymbol{v})$ with $\tau^{S_{i-1}}(\boldsymbol{v}) \wedge \tau^{r_i}(\boldsymbol{v})$ in (19) for all possible choices of $\tau^{r_i}(\boldsymbol{v})$. Hence, when we show

$$K(\mathcal{M}, \chi(Q)) = 1 \qquad (20)$$

for $Q = \neg\exists$, or $Q = \exists^{\omega}$, we have shown (18) for the same Q and all $\tau(\boldsymbol{v}) \supseteq \tau^{S_{i-1}}(\boldsymbol{v})$.

To show (20), first consider the case that

$$\mathcal{M} \models \forall \boldsymbol{v}(\tau^{S_{i-1}}(\boldsymbol{v}) \rightarrow \neg\exists w \sigma^{S_{i-1}, w}(\boldsymbol{v}, w)). \qquad (21)$$

In this case (20) holds with $Q = \neg\exists$, and we are done. If, in particular, $\mathcal{M} \models \neg\exists \boldsymbol{v} \tau^{S_{i-1}}(\boldsymbol{v})$, then (20) holds for both $Q = \neg\exists$ and $Q = \exists^{\omega}$.

If (21) does not hold, then because $\mathcal{M} \models \Phi^N_{\text{Ext}}(S_{i-1})$,

$$\mathcal{M} \models \exists^{\omega} \boldsymbol{v} \tau^{S_{i-1}}(\boldsymbol{v}) \qquad (22)$$

(by lemma 4.4), and

$$\mathcal{M} \models \forall \boldsymbol{v}(\tau^{S_{i-1}}(\boldsymbol{v}) \rightarrow \exists^{\omega}(\sigma^{S_{i-1}, w}(\boldsymbol{v}, w)). \qquad (23)$$

Now consider a fixed tuple $\boldsymbol{d}$ with $\mathcal{M} \models \tau^{S_{i-1}}(\boldsymbol{d})$. By (23) there exist $e_1, e_2, \ldots \in D$ s.t. $\mathcal{M} \models \sigma^{S_{i-1}, w}(\boldsymbol{d}, e_i)$ for $i \in \omega$. In the conjunction $\sigma^{r_i, w}(\boldsymbol{d}, e_i)$ let $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_k$ be the n-tuples over $(\boldsymbol{d}, e_i)$ s.t. $r_i(\boldsymbol{f}_j)$ appears in $\sigma^{r_i, w}(\boldsymbol{d}, e_i)$, and $g_1, \ldots, g_l$ be the tuples for which $\neg r_i(g_h)$ appears in $\sigma^{r_i,w}(d, e_i)$. Then

$$K(\mathcal{M}, \sigma^{r_i,w}(d, e_i)) = \prod_{j=1}^{k} F_{r_i}(f_j)[\mathcal{M}] \prod_{h=1}^{l}(1 - F_{r_i}(g_h)[\mathcal{M}]) =: p. \quad (24)$$

By lemma 4.2, p is independent of the actual choice of d and e_i, as long as $\mathcal{M} \models \tau^{S_{i-1}}(d) \wedge \sigma^{S_{i-1},w}(d, e_i)$.

Again, we now distinguish two cases. First assume that $p = 0$. Then

$$K(\mathcal{M}, \neg\exists w(\sigma^{S_{i-1},w}(d, w) \wedge \sigma^{r_i,w}(d, w)) \geq 1 - \sum_{i \in \omega} K(\mathcal{M}, \sigma^{r_i,w}(d, e_i)) = 1. \quad (25)$$

Now assume $p > 0$. Then

$$K(\mathcal{M}, \exists^{\omega} w(\sigma^{S_{i-1},w}(d, w) \wedge \sigma^{r_i,w}(d, w))) = K(\mathcal{M}, \cap_{j \geq 1} \cup_{i \geq j} \sigma^{r_i,w}(d, e_i)) = 1. \quad (26)$$

The last identity in (26) follows from the Borel-Cantelli lemma, using that for $i \neq j$ $\sigma^{r_i,w}(d, e_i)$ and $\sigma^{r_i,w}(d, e_j)$ are independent in $\mathfrak{A}_{r_i}$ with respect to $K(\mathcal{M}, \cdot)$ (because $\sigma^{r_i,w}(d, e_i)$ and $\sigma^{r_i,w}(d, e_j)$ do not share any common r_i-atom).

By our observation following (24), whether (25) or (26) holds for d does not depend on the particular choice of d, as long as $\mathcal{M} \models \tau^{S_{i-1}}(d)$. Taking the conjunction over the countably many d, we can finally say:

$$K(\mathcal{M}, \chi(\neg\exists)) = 1 \quad (\text{if } p = 0 \text{ in } (24)), \text{ or}$$
$$K(\mathcal{M}, \chi(\exists^{\omega})) = 1 \quad (\text{if } p > 0 \text{ in } (24)).$$

□

The following lemma is closely related to the fact that in the preceding proof the question of whether to include $\chi(\neg\exists)$ or $\chi(\exists^{\omega})$ into the extension theory for S_i was essentially reduced to the question of whether p in (24) is positive. We now make this reduction somewhat more explicit, by showing that we can "decide" Φ^N_{Ext} via computation of certain probabilities.

Lemma 4.6 Let $N, \Phi^N_{\text{Ext}}(S_i)$ be as in theorem 4.5. Let $\tau(v) \subset \sigma(v, w)$ be S_i-types. Let $d, e \subset \omega$ be arbitrary tuples with $|\, d \,| = |\, v \,|$, $|\, e \,| = |\, w \,|$, and such that d satisfies the equality constraints in $\tau(v)$, and (d, e) satisfies the equality constraints in $\sigma(v, w)$. For

$$\lambda(Q) :\equiv \forall v(\tau(v) \rightarrow Qw\sigma(v, w)) \; (Q \in \{\neg\exists, \exists^{=1}, \exists^{\omega}\})$$

we then have

(a) $\Phi^N_{\text{Ext}} \models \lambda(\exists^{=1})$ iff $\sigma(v, w) \models w \sqsubseteq v$;

(b) $\Phi^N_{\text{Ext}} \models \lambda(\neg\exists)$ iff $P^N_{\omega}(\tau(d)) = 0$ or $P^N_{\omega}(\sigma(d, e)) = 0$;

(c) $\Phi^N_{\text{Ext}} \models \lambda(\exists^{\omega})$ iff $P^N_{\omega}(\tau(d)) = 0$ or $P^N_{\omega}(\sigma(d, e)) > 0$.

Proof: Part (a) is already covered by lemma 4.4. For parts (b) and (c) first observe that $\Phi^N_{\text{Ext}}(S_i) \models \lambda(\neg\exists) \wedge \lambda(\exists^{\omega})$ iff $\Phi^N_{\text{Ext}}(S_i) \models \neg\exists v\tau(v)$. By symmetry, we have that $P^N_{\omega}(\tau(d)) = P^N_{\omega}(\tau(d'))$ for all of the countably many tuples d' that satisfy the equality constraints in $\tau(v)$. With $P^N_{\omega}(\exists v\tau(v)) = P^N_{\omega}(\cup_{d'} \tau(d'))$ we thus get $P^N_{\omega}(\tau(d)) = 0$ iff $P^N_{\omega}(\neg\exists v\tau(v)) = 1$ iff $\Phi^N_{\text{Ext}}(S_i) \models \neg\exists v\tau(v)$.

By analogous symmetry and countability arguments we get $P^N_{\omega}(\sigma(d, e)) > 0$ iff $P^N_{\omega}(\neg\exists vw\sigma(v, w)) = 0$ iff $P^N_{\omega}(\lambda(\neg\exists)) = 0$ iff $P^N_{\omega}(\lambda(\exists^{\omega})) = 1$ iff $\Phi^N_{\text{Ext}}(S_i) \models \lambda(\exists^{\omega})$. □

4.2 COMPUTING THE PROBABILITIES

We now show how the insights gained in the preceding section into the structure that, according to P^N_{ω}, an infinite structure $\mathcal{M}$ will almost certainly have enable us to compute probabilities in P^N_{ω}. The central insight is provided by lemma 4.2. According to that lemma, when we evaluate $F_{r_i}(d)[\mathcal{M}]$ for some canonical $\mathcal{M}$, we only need to know the S_i-type of d. This means that we can in effect replace F_{r_i} by a fundamentally simpler formula $F^*_{r_i}$:

Theorem 4.7 Let N be an admissible relational Bayesian network over $S = \{r_1, \ldots, r_k\}$ with *Pa*$(r_i) \subseteq \{r_1, \ldots, r_{i-1}\}$ $(i = 1, \ldots, k)$. There exists a relational Bayesian network N^* over S with the following properties

(i) *Pa*$(r_i) \subseteq \{r_1, \ldots, r_{i-1}\}$ in N^*.

(ii) Every probability formula $F^*_{r_i}$ in N^* is of the form

$$F^*_{r_i}(v) = \begin{cases} q_1 & \tau_1(v) \\ \vdots & \\ q_m & \tau_m(v) \end{cases} \quad (27)$$

with $q_h \in [0, 1]$ $(h = 1, \ldots, m)$, and the $\tau_h(v)$ are a complete list of the *Pa*(r_i)-types of v.

(iii) $P^{N^*}_{\omega} = P^N_{\omega}$.

Given N, the network N^* can be computed effectively.

The salient feature of the probability formulas $F^*_{r_i}$ is that they are essentially combination function free (even though, in proper syntax, the distinction by cases in (27) would by encoded by some "benign" combination functions). Specifically, the formula $F^*_{r_i}(v)$ does not contain any variables w

other than $\boldsymbol{v}$. In that sense, theorem 4.7 can be understood as a kind of quantifier elimination result, that allows us to get rid of the quantification over $\boldsymbol{w}$ by combination functions *comb*$\{\cdot \mid \boldsymbol{w};\cdot\}$. In practical terms, this means that the probability of a statement $\phi(\boldsymbol{d})$ can be computed without considering any elements other than $\boldsymbol{d}$. This is spelled out in the following corollary, which is of central interest in itself, and also plays an important role in the proof of the theorem.

Corollary 4.8 Let N be an admissible relational Bayesian network. Let $\phi(\boldsymbol{d})$ be a boolean expression over ground S-atoms. The probability $P^N_\omega(\phi(\boldsymbol{d}))$ then is computable.

Proof: First observe that the computation of $P^N_\omega(\phi(\boldsymbol{d}))$ can be reduced to finitely many computations of probabilities $P^N_\omega(\sigma(\boldsymbol{d}))$ for S-types $\sigma(\boldsymbol{v})$. The S-type $\sigma(\boldsymbol{v})$ can be written as a conjunction $\wedge^k_{i=1}\sigma_i(\boldsymbol{v})$ of S_i-types σ_i. For $\boldsymbol{d}' \subseteq \boldsymbol{d}$ let $\sigma_i(\boldsymbol{d}')$ be the S_i-type of $\boldsymbol{d}'$ implied by $\sigma(\boldsymbol{d})$. Then

$$P^N_\omega(\sigma(\boldsymbol{d})) =$$

$$\prod_{i=1}^{k} P^N_\omega(\sigma_i(\boldsymbol{d}) \mid \sigma_{i-1}(\boldsymbol{d})) =$$

$$\prod_{i=1}^{k} \prod_{\{\boldsymbol{d}'\subseteq \boldsymbol{d} \mid \sigma_i(\boldsymbol{d})\models r_i(\boldsymbol{d}')\}} F_{r_i}(\boldsymbol{d}')[\sigma_{i-1}(\boldsymbol{d}')] \cdot \prod_{\{\boldsymbol{d}'\subseteq \boldsymbol{d} \mid \sigma_i(\boldsymbol{d})\models \neg r_i(\boldsymbol{d}')\}} (1 - F_{r_i}(\boldsymbol{d}')[\sigma_{i-1}(\boldsymbol{d}')]).$$

The terms $F_{r_i}(\boldsymbol{d}')[\sigma_{i-1}(\boldsymbol{d}')]$, here stand for $F_{r_i}(\boldsymbol{d}')[\mathcal{M}]$ for an arbitrary $\mathcal{M} \models \Phi^N_{\text{Ext}} \wedge \sigma_{i-1}(\boldsymbol{d}')$. Using the network N^* we can determine $q_h = F_{r_i}(\boldsymbol{d}')[\sigma_{i-1}(\boldsymbol{d}')]$ by looking up the *Pa(r$_i$)*-type τ_h that is implied by σ_{i-1}. □

Proof of theorem 4.7: We show how to effectively construct probability formulas $F^*_{r_i}$ over $\{r_1, \dots, r_{i-1}\}$ such that for all $\mathcal{M} \models \Phi^N_{\text{Ext}}(S_{i-1})$, and all $\boldsymbol{d} \in \omega^{|r_i|}$ we get

$$F_{r_i}(\boldsymbol{d})[\mathcal{M}] = F^*_{r_i}(\boldsymbol{d})[\mathcal{M}]. \tag{28}$$

With theorem 4.5 it then follows that the network N^* defined by the probability formulas $F^*_{r_i}$ defines the same distribution as the original network N.

For the construction of $F^*_{r_i}$ assume that $F^*_1, \dots, F^*_{r_{i-1}}$ already have been constructed (the base case F^*_1, again, is a simpler variation of the induction step, and is here omitted). We define $F^*_{r_i}$ by induction on the structure of F_{r_i}.

The first three cases – F_{r_i} being a constant, an indicator function, or a convex combination are quite straightforward. We therefore turn directly to the case $F_{r_i}(\boldsymbol{v}) \equiv$ *comb*$\{F(\boldsymbol{v},\boldsymbol{w}) \mid \boldsymbol{w}; c(\boldsymbol{v},\boldsymbol{w})\}$ (to simplify matters slightly, we here only consider the case of a single probability formula within the combination function. The extension to the general case is straightforward). By induction hypothesis, we can assume that $F(\boldsymbol{v},\boldsymbol{w})$ is in the form (27) with values $p_1, \dots, p_l$ for types $\sigma_1(\boldsymbol{v},\boldsymbol{w}), \dots, \sigma_l(\boldsymbol{v},\boldsymbol{w})$ in some vocabulary $S^* \subseteq S_{i-1}$. We now define *Pa(r$_i$)* to be the smallest subset of S_{i-1} that contains S^*, and for which Φ^N_{Ext} provides a complete set of extension axioms. In concrete terms, this is S_j for $j := max\{h \mid r_h \in S^*\}$.

Now, let $\tau_h(\boldsymbol{v})$ be a *Pa(r$_i$)−type*. We need to find the value q_h such that

$$q_h = F_{r_i}(\boldsymbol{d})[\mathcal{M}] \tag{29}$$

for all $\mathcal{M} \models \Phi_{\text{Ext}}$ and $\boldsymbol{d} \in \omega^{|r_i|}$ with $\mathcal{M} \models \tau_h(\boldsymbol{d})$. By lemma 4.2 we know that q_h only depends on τ_h, but not on $\boldsymbol{d}$ or $\mathcal{M}$ (note that, while F_{r_i} is an S^*-probability formula, we really do need to fix the *Pa(r$_i$)*[$\supseteq S^*$]-type of $\boldsymbol{d}$ in order to insure this invariance, because Φ^N_{Ext} usually will not include an extension theory for S^*).

The first case to be considered for the computation of q_h is whether $\Phi^N_{\text{Ext}} \models \neg\exists \boldsymbol{v}\tau_h(\boldsymbol{v})$. In that case an arbitrarily chosen value for q_h will satisfy (29). By lemma 4.6, we can determine whether $\Phi^N_{\text{Ext}} \models \neg\exists \boldsymbol{v}\tau_h(\boldsymbol{v})$ by computing $P^N_\omega(\tau_h(\boldsymbol{d}))$ for some test tuple $\boldsymbol{d}$. This probability is determined by the subnetwork N_{i-1} containing all the S_{i-1}-nodes. By induction hypothesis (for the outer induction on the network structure), N^*_{i-1} already has been constructed, so that by corollary 4.8, $P^N_\omega(\tau_h(\boldsymbol{d}))$ can be computed. If $P^N_\omega(\tau_h(\boldsymbol{d})) = 0$, we set, e.g., $q_h := 0$, and are done with τ_h.

If $P^N_\omega(\tau_h(\boldsymbol{d})) > 0$ we continue as follows. By definition, $q_h = F_{r_i}(\boldsymbol{d})[\mathcal{M}] = comb\, A$ for

$$A = \{F(\boldsymbol{d},\boldsymbol{e}) \mid \boldsymbol{e}; c(\boldsymbol{d},\boldsymbol{e})\}$$

(with $\boldsymbol{d}$ and $\mathcal{M}$ as in (29)). Thus, we need to determine A. We do this by computing for each *Pa(r$_i$)*-type $\rho(\boldsymbol{v},\boldsymbol{w})$ the contribution of those elements $\boldsymbol{e}$ to A that satisfy $\mathcal{M} \models \rho(\boldsymbol{d},\boldsymbol{e})$. Since $\boldsymbol{d}$ was assumed to be of type τ_h, we only need to consider types ρ that extend τ_h. Also, because of the restriction to $\boldsymbol{e}$ with $c(\boldsymbol{d},\boldsymbol{e})$, only types $\rho(\boldsymbol{v},\boldsymbol{w})$ with $\rho(\boldsymbol{v},\boldsymbol{w}) \models c(\boldsymbol{v},\boldsymbol{w})$ are relevant (note that either $\rho(\boldsymbol{v},\boldsymbol{w}) \models c(\boldsymbol{v},\boldsymbol{w})$, or $\rho(\boldsymbol{v},\boldsymbol{w}) \models \neg c(\boldsymbol{v},\boldsymbol{w})$). There is at most one such type $\rho_0(\boldsymbol{v},\boldsymbol{w})$ with $\rho_0(\boldsymbol{v},\boldsymbol{w}) \models \boldsymbol{w} \sqsubseteq \boldsymbol{v}$. For that type, there exists exactly one $\boldsymbol{e}_0$ with $\rho_0(\boldsymbol{d},\boldsymbol{e}_0)$, and ρ_0 contributes a single copy of $F(\boldsymbol{d},\boldsymbol{e}_0)[\mathcal{M}]$ to A. The value of $F(\boldsymbol{d},\boldsymbol{e}_0)[\mathcal{M}]$ is given in our representation of F^* by p_j corresponding to the S^*-type $\sigma_j(\boldsymbol{v},\boldsymbol{w}) = \rho_0(\boldsymbol{v},\boldsymbol{w}) \upharpoonright S^*$.

For $\rho \not\models \boldsymbol{w} \sqsubseteq \boldsymbol{v}$, by lemma 4.6, we have that $\mathcal{M} \models \neg\exists \boldsymbol{w}\rho(\boldsymbol{d},\boldsymbol{w})$ iff $P^N_\omega(\rho(\boldsymbol{d},\boldsymbol{e})) = 0$, and $\mathcal{M} \models \exists^\omega \boldsymbol{w}\rho(\boldsymbol{d},\boldsymbol{w})$ iff $P^N_\omega(\rho(\boldsymbol{d},\boldsymbol{e})) > 0$ for any test tuple $\boldsymbol{e}$. As above, the probabilities on the right hand sides of these equivalences can be computed in the already constructed network N^*_{i-1}. If $P^N_\omega(\rho(\boldsymbol{d},\boldsymbol{e})) > 0$, the type ρ contributes to A ω copies of the value p_j assigned by F^* to arguments of type $\rho \upharpoonright S^*$. If $P^N_\omega(\rho(\boldsymbol{d},\boldsymbol{e})) = 0$, the type ρ does not contribute to A.

After performing these computations for all relevant types ρ, we obtain a representation of A of the form $A = \{p_{h_1} : \lambda_1, \ldots, p_{h_m} : \lambda_m\}$. where the $p_{h_j} \in \{p_1, \ldots, p_l\}$, and $\lambda_j \in \{1, \omega\}$ with $\lambda_j = 1$ for at most one j. Finally, we compute $q_h := comb\ A$ □

If we apply the transformation here described to our network from example 2.9, the probability formulas of r_1 and r_2 remain essentially unchanged. In the construction of $F^*_{r_3}(v)$ we will first obtain the new parent-set $Pa(r_3) = \{r_1, r_2\}$. The values q_j in the representation (27) are computed as $q_j = 1$ for $\{r_1, r_2\}$-types $\tau_j(v)$ with $\tau_j(v) \models r_1(v)$, and $q_j = 0$ for $\tau_j(v) \models \neg r_1(v)$. This means that in an additional step we can simplify $F^*_{r_3}$ to

$$F^*_{r_3}(v) = \begin{cases} 1 & r_1(v) \\ 0 & \neg r_1(v). \end{cases}$$

This illustrates that the transformation $N \mapsto N^*$ not only replaces probability formulas, but in the process also changes the underlying network structure.

5 Infinite Domains: Recursive Networks

We now turn to the question of what part of the results of sections 3 and 4 carry over to recursive RBNs. As might be expected, in the case $D = \omega$ introduction of recursion causes much more fundamental problems than was the case for finite domains.

The first problem we are faced with is the existence of semantics for a recursive RBN. Recall that we identified as one main advantage of (non-recursive) RBNs their semantic clarity, particularly the fact that simply adhering to the syntactic rules for the construction of an RBN guarantees the user that exactly one distribution P^N_D will be defined. This is true both for finite and infinite D. Turning to recursive RBNs, in section 2.2 we found that for finite domains the existence of semantics no longer is guaranteed, but that a decision procedure for the well-definedness of P^N_D exists in checking the well-foundedness of the relations $\prec_r$ defined by formulas *r-pa(v, w)*. One can show that in the case $D = (\omega, s)$ well-foundedness of $\prec_r$ on $\omega^{|r|}$ still enables us to define P^N_ω (though the construction is more complicated than the one in section 3). It is unclear, however, that it is possible, in general, to decide whether a given formula *r-pa(v, w)* defines a well-founded relation $\prec_r$ on $\omega^{|r|}$. While the problem of decidability of well-foundedness of $\prec_r$ is still open, the following theorem tells us that even in the (unlikely) event of a positive solution, not too much would be gained, because the problem of ultimate interest, computation of probabilities, will still be unsolvable.

Theorem 5.1 There does not exist an algorithm that, given a recursive RBN N over S with defined semantics P^N_ω as input, enumerates all probabilities $P^N_\omega(r(\mathbf{d}))$ for $r \in S$, $\mathbf{d} \in \omega^{|r|}$.

The theorem can be paraphrased as: there does not exist a sound and complete proof system for recursive RBNs with well-defined semantics over infinite domains.

Proof: We reduce the validity problem of first-order sentences in arithmetic to the computation of probabilities $P^N_\omega(r(\mathbf{d}))$. In order to fit into our relational framework, we here code sentences in arithmetic by using a relational vocabulary with unary relation symbols r_0, r_1 (rather than constant symbols 0,1), and ternary relation symbols $r_+, r_.$ (rather than binary function symbols $+, \cdot$). The standard model $\mathcal{N}$ for the vocabulary $S_{\text{ar}} := \{r_0, r_1, r_+, r_.\}$ then is ω with $I(r_0) = \{0\}$, $I(r_1) = \{1\}$, $I(r_+) = \{(d, e, f) \mid d + e = f\}$, and $I(r_.) = \{(d, e, f) \mid d \cdot e = f\}$.

Given an S_{ar}-sentence ϕ, we construct a network N_ϕ containing a unary relation r_ϕ, so that P^N_ω is defined, and $P^N_\omega(r_\phi(1)) = 1$ iff $\mathcal{N} \models \phi$. The non-enumerability of P^N_ω then follows from the non-enumerability of the theory of $\mathcal{N}$.

N_ϕ is constructed as follows. We first define a network N_{ar} over S_{ar} that assigns probability 1 to $\mathcal{N}$. This is achieved by the probability formulas

$$F_{r_0}(v) \equiv 1 - max\{1 \mid w; s(w) = v\}$$

$$F_{r_+}(v, w, z) \equiv \begin{cases} 1 & (r_0(w) \wedge v = z) \vee \\ & r_+(v, s^{-1}(w), s^{-1}(z)) \\ 0 & \text{else} \end{cases}$$

and similar formulas for F_{r_1} and $F_{r_.}$. Here a somewhat loose syntax has been used for F_{r_+}. It is quite straightforward, though, to transform it into a probability formula proper. The dependency relations $\prec_{r_+}$ and $\prec_{r_.}$ are well-founded, and $P^{N_{\text{ar}}}_\omega(\mathcal{N}) = 1$, as desired.

Given the S_{ar}-sentence ϕ, according to lemma 2.6, we can define a (non-recursive) probability formula $F_\phi(v)$, so that for all $d \in \omega$: $F_\phi(d)[\mathcal{N}] = 1$ iff $\mathcal{N} \models \phi$; $F_\phi(d)[\mathcal{N}] = 0$ else. Adding a node r_ϕ labeled with F_ϕ to the network N_{ar} yields a network N_ϕ that, if $\mathcal{N} \models \phi$, places probability 1 on the r_ϕ-extension $\mathcal{N}_\phi$ of $\mathcal{N}$ with $\mathcal{N}_\phi \models \forall v r_\phi(v)$, and if $\mathcal{N} \models \neg\phi$, places probability 1 on the r_ϕ-extension $\mathcal{N}_{\neg\phi}$ of $\mathcal{N}$ with $\mathcal{N}_{\neg\phi} \models \forall v \neg r_\phi(v)$. Hence, $P^{N_\phi}_\omega(r_\phi(1)) = 1$ iff $\mathcal{N} \models \phi$. □

While theorem 5.1 is formulated as a result for recursive RBNs, arguments similar to the one used in its proof clearly could be applied to other probabilistic representation systems for distributions over infinite, structured domains. Particularly systems like that of Ngo and Haddawy (1995) that use Herbrand universes as underlying domains (which in the special case of a single constant and a single unary function symbol is essentially the same as (ω, s)), provide a potential basis for carrying out the same argument. Exact results along these lines, however, first require a more detailed analysis of the first-order reasoning capabilities within these systems, particularly of the question whether they give rise to analogues of our lemma 2.6.

6 RELATED WORK AND CONCLUSION

Apart from the work by Ngo and Haddawy (1995) already mentioned, examples of previously proposed probabilistic representation and inference systems over infinite domains include work on probabilistic context free grammars (Pynadath & Wellman 1996) and stochastic programs (Koller, McAllester & Pfeffer 1997). In both these frameworks, distributions are defined over richly structured domains (labeled trees, essentially). These works differ in their semantic intent somewhat from the one of Ngo and Haddawy, and the one presented here: they, like standard Bayesian networks, are models of attributes of randomly sampled individuals, only that individuals now come from an infinite domain of distinguishable objects. This is to be contrasted with our objective of modeling random relations between multiple objects (taken from a uniform domain). On a sufficiently high level of abstraction, of course, this distinction is not very strict, since in our approach we can also view a relation between elements of the domain as an attribute of a randomly sampled complete structure.

For none of the frameworks mentioned here, decision procedures for general, complex queries have been given (in the particular rich system of Koller et al. (1997) it is moreover clear that none can exist). Main objective of the present paper has been to show that in the case of RBNs such a decision procedure exists. This was achieved by showing that infinite random structures, as generated by an RBN, with probability one will possess certain structural properties, and that therefore every distribution defined over ω by a network N can also be defined by a network N^* of a particularly simple form. The main, non-trivial, component of inference from N for domain ω is the transformation of N to N^*. Once N^* has been determined, computation of $P_\omega^{N^*}(\phi(\boldsymbol{d}))$ for some query $\phi(\boldsymbol{d})$ proceeds without any reference to the underlying domain.

Acknowledgments

Most of this work was conducted during a visit to Stanford University made possible by Daphne Koller. This work was funded in part by DARPA contract DACA76-93-C-0025, under subcontract to Information Extraction and Transport, Inc.

References

Bacchus, F., Grove, A. J., Halpern, J. Y. & Koller, D. (1997), 'From statistical knowledge bases to degrees of belief', *Artificial Intelligence* **87**, 75–143.

Breese, J. S. (1992), 'Construction of belief and decision networks', *Computational Intelligence* **8**(4), 624–647.

Haddawy, P. (1994), Generating bayesian networks from probability logic knowledge bases, *in* 'Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence'.

Jacobs, K. (1978), *Measure and Integral*, New York, Academic Press.

Jaeger, M. (1997), Relational bayesian networks, *in* 'Proceedings of UAI-97'.

Jaeger, M. (1998), Convergence results for relational bayesian networks, *in* 'Proceedings of the Thirteenth Annual IEEE Symposium on Logic in Computer Science (LICS-98)'. To appear.

Koller, D., McAllester, D. & Pfeffer, A. (1997), Effective bayesian inference for stochastic programs, *in* 'Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97)', pp. 740–747.

Ngo, L. & Haddawy, P. (1995), Probabilistic logic programming and bayesian networks, *in* 'Algorithms, Concurrency and Knowledge (Proceedings ACSC95)', Springer Lecture Notes in Computer Science 1023, pp. 286–300.

Pearl, J. (1988), *Probabilistic reasoning in intelligent systems : networks of plausible inference*, The Morgan Kaufmann series in representation and reasoning, rev. 2nd pr. edn, Morgan Kaufmann, San Mateo, CA.

Pynadath, D. & Wellman, M. (1996), Generalized queries on probabilistic context-free grammars, *in* 'Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)'.

Wellman, M. P., Breese, J. S. & Goldman, R. P. (1992), 'From knowledge bases to decision models', *The Knowledge Engineering Review* **7**(1), 35–53.

Geometric Foundations for Interval-Based Probabilities

Vu Ha and Peter Haddawy
Decision Systems and Artificial Intelligence Lab
Dept. of EE&CS
University of Wisconsin-Milwaukee
Milwaukee, WI 53201
{*vu,haddawy*} *@cs.uwm.edu*

Abstract

The need to reason with imprecise probabilities arises in a wealth of situations ranging from pooling of knowledge from multiple experts to abstraction-based probabilistic planning. Researchers have typically represented imprecise probabilities using intervals and have developed a wide array of different techniques to suit their particular requirements. In this paper we provide an analysis of some of the central issues in representing and reasoning with interval probabilities. At the focus of our analysis is the probability cross-product operator and its interval generalization, the *cc-operator*. We perform an extensive study of these operators relative to manipulation of sets of probability distributtions. This study provides insight into the sources of the strengths and weaknesses of various approaches to handling probability intervals. We demonstrate the application of our results to the problems of inference in interval Bayesian networks and projection and evaluation of abstract probabilistic plans.

1 INTRODUCTION

In recent years the use of imprecise probabilities has found application in a wealth of areas. Imprecise probabilities can be used to facilitate elicitation when the available domain knowledge is insufficient to specify exact probabilities [10, 22], or when eliciting knowledge from multiple experts [20]. They may result from the abstraction of more detailed probabilistic models [2, 13], and are useful in studying sensitivity and robustness in probabilistic inference, e.g. in Bayesian networks [4]. The use of imprecise probabilities is even advocated as an alternative representation of belief by researchers who do not feel comfortable with the paradigm of *Strict Bayesianism* which requires exact probabilistic models [19, 18, 27].

While several methods exist to represent imprecise probabilities, representation using intervals is the most commonly used approach. Researchers working on the problems mentioned in the previous paragraph have developed a number of techniques for handling probability intervals. But to date we lack a uniform and comprehensive study of the central issues concerning representation and manipulation of probability intervals. In this paper, we make the first steps towards such a study. Our approach originates from the observation that the probability cross-product operator lies in the hearts of numerous computations such as probabilistic plan projection, expected utility computation, and Bayesian network propagation. We present an interval generalization of this operator, called the *cc-operator* [1], and provide a throrough analysis of some key properties of the cc-operator relative to manipulation of sets of probability distributions. We then show how the cc-operator can be substituted for the probability cross-product to produce interval versions of plan projection and Bayesian network propagation algorithms. Our Bayesian network propagation algorithm is based on Dechter's Bucket Elimination algorithm [6]. We draw upon our theoretical analysis of the cc-operator to produce efficient versions of these generalized algorithms. Our approach rests on the well-developed area of finite convex geometry. Thus our results are applicable to problems with discrete finite probability distributions.

The rest of this paper is organized as follows. We start Section 2 with a quick review of several convex geometric concepts most relevant to our analysis. We then de-

[1] "cc" stands for "convex combination". It was originally called *affine-operator* in [13]. We adopt this new term in accordance with standard convex geometry terminology.

fine and analyze the cc-operator. In Section 3 we define the concept of cc-trees, which are data structures that have cc-operators as basic building blocks. These data structures, in particular their interval and probability versions, icc-trees and pcc-trees, will be used throughout our analysis. In Section 4, we present the theoretical foundations of an abstraction-based probabilistic planner, DRIPS [14], where the cc-operator plays a fundamental role. In Section 5, we use the concept of icc-trees to develope an interval version of Dechter's Bucket Elimination algorithm for propagation in interval Bayesian networks. For both plan projection and Bayesian network propogation, the probability bounds computed by our algorithms are correct but not tight. We suggest an explanation for the difficulty of computing tight probabilistic bounds in the above problems. In Section 6, we return to analyzing the cc-operator. In particular, we investigate pcc-trees as a new approach to represent convex sets of probability distributions. We show that the class of pcc-trees is identical to the class of polytope-like sets of probability distributions, and that in a special case, Dempster-Shafer belief functions, when viewed as lower envelopes of sets of probability distributions, can be represented by 2-level pcc-trees in at least two different ways. In Section 7, we address the issue of evidential updating with pcc-trees. The in-depth study of the cc-operator provides an insightful explanation of why updating with pcc-trees in general and with belief functions in particular is difficult. The latter issue has been a subject of extensive study, and our analysis adds a new perspective to it.

2 PROBABILITY CROSS-PRODUCT AND THE CC-OPERATOR

We start with a brief introduction of some basic concepts of convex geometry. For more details, see [11].

The *d-dimension Euclidean Space* is the d-dimension vector space $\Re^d$ equipped with the inner product $\langle\rangle$, which is defined for any pair of points $x, y \in \Re^d$ as: $\langle x, y\rangle = \sum_{i=1}^{d} x_i y_i$. A *convex combination* of the points $x_1, \ldots, x_n \in \Re^d$ is a linear combination $\lambda_1 x_1 + \cdots + \lambda_n x_n$, where $\lambda_i \geq 0, i = 1, \ldots, n$, and $\lambda_1 + \cdots + \lambda_n = 1$. The coefficients λ_i are called *convex coefficients*, and the sum is denoted by $\overset{cc}{\sum}$. This sum is also called the *probability cross-product* of 2 vectors whose components are λ_i and x_i. For any set $K \subseteq \Re^d$, the *convex hull* of K, denoted by $\text{conv}(K)$, is the set of all convex combinations of the points from K. A set $K \subseteq \Re^d$ is said to be *convex* if $K = \text{conv}(K)$. A mapping ϕ from a convex set $K \subseteq \Re^d$ to $\Re^e$ is said to be a *convex mapping* if it preserves convex combinations, i.e. $\phi(\overset{cc}{\sum}{}_{i=1}^{n} \lambda_i x_i) = \overset{cc}{\sum}{}_{i=1}^{n} \lambda_i \phi(x_i)$, for all points $x_i \in K$ and convex coefficients λ_i. A convex mapping always maps convex sets into convex sets. The convex hull of a finite set of points is called *a polytope*. Given a polytope K, the smallest set of points whose convex hull is equal K is called the set of *vertices* of the polytope. The polytope whose vertices are the points $(1, 0, \ldots, 0), (0, 1, \ldots, 0), \ldots, (0, 0, \ldots, 1)$ is exactly the set $\mathcal{S}$ of all probability distributions over a sample space Ω of cardinality d and is called the *probability simplex*. For example, the probability distribution P over the sample space $\Omega = \{s_1, s_2, \ldots, s_d\}$ corresponds to the point $(P(s_1), P(s_2), \ldots, P(s_d))$ in $\mathcal{S}$.

We now define the generalized version of the probability cross-product, the cc-operator. In the rest of the paper, an interval is implicitly understood as a closed subinterval of $[0, 1]$.

Definition 1 *The* cc-operator $\otimes$ *defined by an interval vector* $\vec{\Lambda} = (\Lambda_1, \ldots, \Lambda_n)$ *is the function that maps a vector* $\vec{w} = (w_1, \ldots, w_n)$ *of sets of points in* $\Re^d$ *to the set* $\vec{\Lambda} \otimes \vec{w}$ *that consists of all points of the form* $\vec{\lambda} \otimes \vec{x} = \overset{cc}{\sum}{}_{i=1}^{n} \lambda_i x_i$, *where* $\lambda_i \in \Lambda_i, x_i \in w_i, i = 1, \ldots, n$. *We sometimes write* $\vec{\Lambda} \otimes \vec{w}$ *as* $\overset{cc}{\sum}{}_{i=1}^{n} \Lambda_i w_i$. *In the case when* w_i *are also intervals, we call this sum the* icc-operator *sum (letter "i" stands for "interval") and denote it by* $\overset{icc}{\sum}$.

The cc-operator sum $\vec{\Lambda} \otimes \vec{w}$ is a special case of the Minkowski sum, a familiar notion in convex geometry. The difference between the two sums is that the coefficients λ_i in the Minkowski sum are arbitrary. Below are a few properties of the cc-operator, which constitute the basis of our framework.

Fact 1 *Any probability distribution* $P \in \mathcal{S}$ *can be expressed as a cc-operator sum as* $P = (P(s_1), \ldots, P(s_d)) \otimes (s_1, \ldots, s_d)$, *where* s_i *represents the ith vertex of the probability simplex.*

Fact 2 *If* w_i *are sets of probability distributions, then* $\vec{\Lambda} \otimes \vec{w}$ *is also a set of probability distributions. In other words,* $\otimes$ is closed on $2^{\mathcal{S}}$.

Fact 3 *If each of the intervals* Λ_i *is the entire interval* $[0, 1]$, *we denote* $\vec{\Lambda} \otimes \vec{w}$ *as* $cc(\vec{w})$, *or* $cc(w_1, \ldots, w_n)$. *If* K *is a polytope with vertices* $\{x_1, \ldots, x_n\}$, *then* $K = conv(\{x_1, \ldots, x_n\}) = cc(\{x_1\}, \ldots, \{x_n\})$.

Fact 4 The cc-operator and convex mappings commute. *Specifically, suppose that* $K \subseteq \Re^d$ *is a convex* *set and $\phi : K \to \Re^e$ is a convex mapping. For any $H \subseteq K$, defined $\phi(H)$ as $\phi(H) = \{\phi(x) | x \in H\}$. Then*

$$\phi\Big(\overset{cc}{\sum}{}_{i=1}^{n} \Lambda_i w_i \Big) = \overset{cc}{\sum}{}_{i=1}^{n} \Lambda_i \phi(w_i).$$

The cc-operator has the following important property, which was first formulated in [13]. For a proof, see [12].

Theorem 1 The cc-operator and convex hull operator commute. *Specifically:*

$$conv\left(\overset{cc}{\sum}{}_{i=1}^{n} \Lambda_i w_i \right) = \overset{cc}{\sum}{}_{i=1}^{n} \Lambda_i conv(w_i).$$

3 CC-TREES, ICC-TREES, PCC-TREES

In this section we introduce the concept of cc-trees and its special cases: icc-trees and pcc-trees. These trees can be viewed as data structures that encode, among other things, sets of real numbers and sets of probability distributions.

A *cc-tree* is a rooted, annotated tree where the branches are annotated with intervals and the nodes are annotated with sets of points in $\Re^d$, with the constraint that for any internal (i.e. non-leaf) node N with children $\{C_1, \ldots, C_k\}$, the set $\mathcal{D}(N)$ that annotates N is the result of applying the cc-operator defined by the intervals Λ_i that annotate the branches (N, C_i) to the sets $\mathcal{D}(C_i)$ that annotate the children C_i: $\mathcal{D}(N) = \overset{cc}{\sum}{}_{i=1}^{k} \Lambda_i \mathcal{D}(C_i)$. Thus, a cc-tree is completely defined by specifying the sets of points associated with the leaves and the intervals associated with the branches. Intuitively, this tree encodes the set of points obtained by applying the cc-operator from the leaves up toward the root.

Cc-trees whose leaves are annotated with intervals are called *icc-trees.* Figure 1 depicts an example of an icc-tree. We now discuss the evaluation of the icc-operator sum $\overset{icc}{\sum}{}_{i=1}^{n} \Lambda_i w_i$, where $\Lambda_i \subseteq [0\ 1]$ and $w_i \subseteq \Re$ are closed intervals. Note that the set of convex sets of real numbers is exactly the set of intervals. It then follows directly from Theorem 1 (cc-operator and convex hull operator commute) that $\overset{icc}{\sum}{}_{i=1}^{n} \Lambda_i w_i$ is also an interval. We show that it is a closed interval by showing how the lower and upper bounds are explicitly obtained. Due to symmetry, we will consider only the upper bound.

Obviously, the upper bound must be achieved by maximizing $\overset{cc}{\sum}{}_{i=1}^{n} \lambda_i u_i$, where $\lambda_i \in \Lambda_i$ and u_i are upper

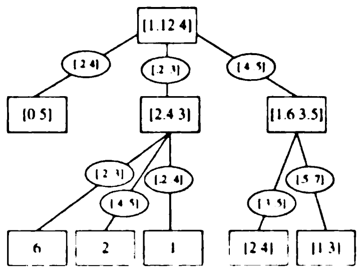


Figure 1: Example of an icc-tree.

bounds of w_i. This is a special case of the continuous knapsack problem: to pack materials from n different categories into a unit size knapsack, where material from category number i has value u_i and weight restriction Λ_i (meaning that the weight must be in the interval Λ_i). The objective is to maximize the total value, which can be achieved using a greedy approach as follows. We first sort the values u_i in *descending* order, breaking ties arbitrarily. Then proceeding in this order, we try to put into the knapsack as much material of the current category as possible, subjected to two constraints: 1) the weight must be in the constraint interval Λ_i and 2) The sum of the already assigned weights must be at most one minus the sum of the lower bounds of the remaining constraint intervals. The first restriction is explicitly mandatory from the description of the problem, while the second one ensures that the lower bound condition will be satisfied in the future [2]. We illustrate this idea through an example.

Example 1 *Let us compute $\overset{icc}{\sum}{}_{i=1}^{3} \Lambda_i w_i$, where $\Lambda_1 = [.2\ .4]$, $\Lambda_2 = [.2\ .3]$, $\Lambda_3 = [.4\ .5]$, $w_1 = [0\ 5]$, $w_2 = [2.4\ 3]$, $w_1 = [1.6\ 3.5]$. Thus $u_1 = 5$, $u_2 = 3$, $u_3 = 3.5$. First, we assign weight to material number 1, which is computed as $\min\{.4, 1 - .2 - .4\} = .4$. The remaining weight is thus $1 - .4 = .6$. We then assign weight to material number 3: $\min\{.5, .6 - .2\} = .4$. And finally, the weight assigned to the last material is $1 - .4 - .4 = .2$, and thus total value is $.4 \times 5 + .4 \times 3.5 + .2 \times 3 = 4$. The lower bound is 1.12. Thus $\overset{icc}{\sum}{}_{i=1}^{3} \Lambda_i w_i = [1.12\ 4]$. This icc-sum is represented in Figure 1 by the tree of depth 1 that contains the root and its 3 children.*

Remark 1 *The complexity of the above algorithm to compute icc-operator sum is dominated by the sorting of the material values, which is $O(n \log n)$.*

[2]Tessem [25] called this step *reinforcement*, and the analogous step in computing the lower bound *annihilation*.

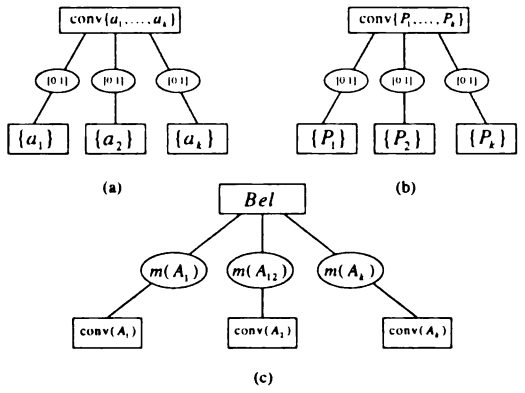


Figure 2: Examples of pcc-trees: (a) The face conv(A) of the probability simplex where $A = \{a_1, \ldots, a_k\} \subseteq \Omega$; (b) A polytope-like set of distributions with vertices $\{P_1, \ldots, P_k\}$; (c) A Dempster-Shafer belief function *Bel* with mass function m and focal elements $\{A_1, \ldots, A_k\}$ represented as a 2-level pcc-tree.

Of our particular interest are cc-trees whose leaves are annotated with the vertices of the probability simplex (or more precisely, sets which contain a single element that is a vertex of the probability simplex). Such trees encode sets of probability distributions and will be called *pcc-trees* (the *"p"* letter stands for "probability").

Figure 2 depicts several pcc-trees. Suppose that $A = \{a_1, \ldots, a_k\} \subseteq \Omega$ is a set of vertices of the probability simplex $\mathcal{S}$. The pcc-tree in Figure 2.a represents conv(A), which is called a *face* of the polytope $\mathcal{S}$. The pcc-tree in Figure 2.b represents a polytope-like sets of probability distributions, which is a polytope with vertices $P_i \in \mathcal{S}$. The pcc-tree in Figure 2.c represents a Dempster-Shafer belief function viewed as the lower envelope of a set of probability distributions. This pcc-tree will be discussed later in Section 6.

4 ABSTRACTION-BASED PROBABILISTIC PLANNING

In the framework of decision-theoretic planning, uncertainty in the state of the world and in the effects of actions are represented with probabilities, and the planner's goals, as well as tradeoffs among them, are represented with utilities. Given this representation, the objective is to find an optimal plan or policy, where optimality is defined as maximizing expected utility.

In most of the existing decision-theoretic planning approaches, the world is represented with a probability distribution over the state space, and actions are represented as stochastic mappings among the states [15, 5, 1, 26]. Given this framing of the problem, all probabilistic and decision-theoretic planners face the burden of computational complexity in seeking an optimal or near-optimal solution. One popular way to address this problem is to use abstraction techniques to guide the search through the plan space and to reduce the cost of plan evaluation. This concept has been applied in Markov process-based planning [1] as well as less structured approaches [15].

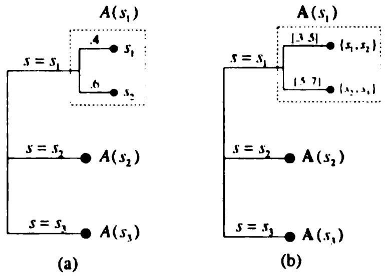


Figure 3: Example of (a) a concrete action and (b) an abstract action.

In the framework of DRIPS, the planning problem is formalized as the problem of searching for the optimal plan through the space of candidate plans, each of which is a sequence of actions. The planner is armed with the capability of understanding and evaluating abstract plans constructed from abstract actions. An abstract action is defined as a mapping from a probability distribution over the state space to a *set* of probability distributions over the state space, and is viewed as a *sound abstraction* of several concrete actions. By evaluating abstract plans, the planner is able to evaluate and eliminate a set of suboptimal plans without explicitly evaluating each member of that set. Herein lies the computational efficiency of DRIPS.

We now present the underlying theoretical framework of DRIPS, where the cc-operator and the derived cc-trees play fundamental roles.

4.1 PROBABILISTIC ACTIONS AND PLANS

A *probabilistic action* A is defined as a function $A : \Omega \to \mathcal{S}$ that maps a state $s \in \Omega$ of the world to a probability distribution $A(s)$ over possible states of the world. For example, the action in Figure 3.a maps state s_1 to the probability distribution $A(s_1)$ that assigns s_1 the probability of .4 and s_2 the probability of .6, and maps states s_2 and s_3 to some other distributions $A(s_2), A(s_3)$ over $\Omega = \{s_1, s_2, s_3\}$. The distributions $A(s), s \in \Omega$ are called the *branches of action A*. This function A is then extended to a function that maps a probability distribution $P \in \mathcal{S}$ to a probability distribution $A(P) = E_P[A(s)] = \overset{cc}{\sum}_{s \in \Omega} P(s)A(s)$, which is essentially the probability cross-product of the distribution P and the action A.

A *probabilistic plan pl* is a finite sequence of probabilistic actions: $pl = \{A_1, \ldots, A_n\}$. Given an initial world $w_{ini} \in \mathcal{S}$ that is a probability distribution over the state space Ω, the *projection* of pl on w_{ini} is the process of determining the final world $w_{final} = pl(w_{ini}) = A_n(\ldots A_1(w_{ini})\ldots) \in \mathcal{S}$.

A *utility function* is a function $u : \Omega \to \Re$ that assigns to each state of the world a real number that indicates the desirability of that state. The expected utility of a plan pl is defined as $E_u(pl) = \overset{cc}{\sum}_{s \in \Omega} w_{final}(s)u(s)$, where $w_{final} = pl(w_{ini})$. It is convenient to extend the definition of the utility function u to $u : \mathcal{S} \to \Re$, where $u(P) = E_P[u(s)] = \overset{cc}{\sum}_{s \in \Omega} P(s)u(s), \forall P \in \mathcal{S}$. Now, the expected utility of a plan pl, executed on an initial world w_{ini} can be written as $u(w_{final}) = u(pl(w_{ini}))$. A plan pl is *optimal* if it has the maximum expected utility among all candidate plans.

Probabilistic actions and utility functions, as functions that have $\mathcal{S}$ as their domains, have the following important property.

Theorem 2 *Probabilistic actions, probabilistic plans, and utility functions are convex mappings and thus commute with the cc-operator (Fact 4).*

Proof: Note that both probabilistic actions and utility functions are first defined as functions over Ω, and then extended by means of expectation to functions over $\mathcal{S}$. Let $\phi : \Re^d \supseteq \mathcal{S} \to \Re^e$ be any function defined and extended in the same way, i.e. $\phi(P) = \overset{cc}{\sum}_{s \in \Omega} P(s)\phi(s), \forall P \in \mathcal{S}$. We show that ϕ is a convex mapping as follows:

$$\begin{aligned}
\phi(\overset{cc}{\sum}{}_{i=1}^{n} \lambda_i P_i) &= \overset{cc}{\sum}{}_{s \in \Omega} \left(\overset{cc}{\sum}{}_{i=1}^{n} \lambda_i P_i \right)(s)\phi(s) \\
&= \overset{cc}{\sum}{}_{s \in \Omega} \left(\overset{cc}{\sum}{}_{i=1}^{n} \lambda_i P_i(s) \right)\phi(s) \\
&= \overset{cc}{\sum}{}_{i=1}^{n} \lambda_i \overset{cc}{\sum}{}_{s \in \Omega} P_i(s)\phi(s) \\
&= \overset{cc}{\sum}{}_{i=1}^{n} \lambda_i \phi(P_i).
\end{aligned}$$

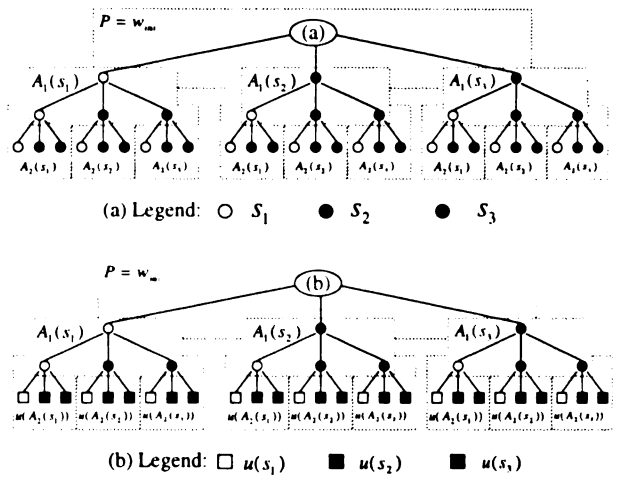


Figure 4: Illustration of (a) projecting and (b) evaluating probabilistic plans.

Thus, probabilistic actions and utility functions are convex mappings. It is easy to see that compositions of convex mappings are also convex mappings, and hence probabilistic plans are also convex mappings. □

Figure 4 illustrates the implications of Theorem 2 on the process of projecting and evaluating probabilistic plans. Consider projecting a 2-action plan $pl = \{A_1, A_2\}$ on the initial world $w_{ini} = P$. The state space Ω consists of three elements $\{s_1, s_2, s_3\}$. The initial world is represented by a pcc-tree with three leaves s_i and three branches $P(s_i)$ (Fact 1). In Figure 4.a, this is the pcc-tree with root (a) and nodes of depth 1. Now consider the world after executing action A_1, namely $A_1(P)$. Since A_1 commutes with the cc-operator defined by P, $A_1(P)$ can be represented by the pcc-tree with root (a) and nodes of depth 1 and 2. We in effect obtained this tree $(A_1(P))$ by replacing the leaves s_i of the pcc-tree P by distributions $A_1(s_i)$. Similarly, we can obtain the pcc-tree representing $A_2(A_1(P))$ by replacing the leaves s_i of the pcc-tree $A_1(P)$ with $A_2(s_i)$. This pcc-tree is the whole tree of depth 3 in Figure 4.a. To compute the expected utility of pl, which is $u(A_2(A_1(P)))$, we can replace the leaves s_i of the pcc-tree representing $A_2(A_1(P))$ with $u(s_i)$ (due to the fact that utility function commute with the cc-operator) and then evaluate the obtained icc-tree (Figure 4.b).

4.2 ABSTRACT ACTIONS AND PLANS

From now on, we will call probabilistic actions *concrete actions*. We generalize concrete actions into *abstract actions*, according to [13], by allowing the probabilities to be interval-values instead of point-values, and the resulting worlds to be sets of states instead of states. This definition is best understood through an example (see Figure 3.b). An abstract action is interpreted as a set of concrete actions each of which is obtained by *instantiation*, a process that replaces the probability intervals with consistent probability numbers, and sets of states with consistent states. The consistency condition for probability numbers means that they must be in the corresponding intervals, and obey the axioms of probability. The consistency conditions for states means that they must be in the corresponding sets. For example, the concrete action A in Figure 3.a is an instantiation of the abstract action $\mathbf{A}$ in Figure 3.b. We denote the instantiation relationship as $A \in \mathbf{A}$. The semantics of an abstract action $\mathbf{A}$ is that it is a function $\mathbf{A} : \mathcal{S} \to 2^{\mathcal{S}}$ that maps a probability distribution $P \in \mathcal{S}$ to a set of probability distributions $\{A(P) | A \in \mathbf{A}\}$.

An *abstract plan* **pl** is a sequence of actions, among which some actions may be abstract actions. An *instantiation plan pl* of an abstract plan **pl** is obtained by replacing each abstract action in **pl** with one of its instantiated actions. We denote this relationship as $pl \in \mathbf{pl}$. The projection of an abstract plan **pl** on an initial world w_{ini} is the process of determining the final world w_{final}, which is defined as: $w_{final} = \{pl(w_{ini}) | pl \in \mathbf{pl}\}$. The expected utility of an abstract plan **pl** is defined as the $u(\mathbf{pl}(w_{ini})) = \{u(pl(w_{ini})) | pl \in \mathbf{pl}\}$.

These notions of abstract actions and plans are introduced in [13] to model sound abstraction, the central concept of DRIPS. An abstract action $\mathbf{A}$ is called a *sound abstraction* of n concrete actions $A_1, \ldots, A_n$ if for any initial world $w_{ini} \in \mathcal{S}$, we have $A_i(w_{ini}) \in \mathbf{A}(w_{ini}), i = 1, \ldots, n$. For example, the abstract action $\mathbf{A}$ in Figure 3.b is a sound abstraction of the action A in Figure 3.a. The use of abstraction allows DRIPS to evaluate a set of concrete plans, represented as a single abstract plan, in a single step, rather than having to evaluate each of the concrete plans in that set individually. This can substantially reduce the time complexity of finding optimal plans, as demonstrated by a number of toy examples as well as successful real-world applications of DRIPS [14].

Evaluating an abstract plan for this purpose amounts to determining the infimum and the supremum of the set $u(\mathbf{pl}(w_{ini})) \subseteq \Re$, which are exactly the endpoints of the interval $\mathrm{conv}(u(\mathbf{pl}(w_{ini})))$. We call this the *expected utility interval* of **pl**. This interval can be approximately computed with the help of the next theorem [13], which is a generalization of Theorem 2.

Theorem 3 *Abstract actions and abstract plans semi-commute with the cc-operator. Specifically, if* $\mathbf{A}$ *is an abstract action or an abstract plan, then*

$$\mathbf{A}(\overset{cc}{\sum}{}_{i=1}^{n} \Lambda_i w_i) \subseteq \overset{cc}{\sum}{}_{i=1}^{n} \Lambda_i \mathbf{A}(w_i).$$

As a consequence of the above theorem, if the initial world w_{ini} is a set of probability distributions [3] that can be written as $w_{ini} = \overset{cc}{\sum} \Lambda_i w_i, w_i \subseteq \mathcal{S}$, then the expected utility interval of an abstract plan **pl**, projected on w_{ini} can be approximated as:

$$\begin{aligned}
\mathrm{conv}(u(\mathbf{pl}(w_{ini}))) &= u(\mathrm{conv}(\mathbf{pl}(w_{ini}))) \\
&= u(\mathrm{conv}(\mathbf{pl}(\overset{cc}{\sum} \Lambda_i w_i))) \\
&\subseteq u(\mathrm{conv}(\overset{cc}{\sum} \Lambda_i \mathbf{pl}(w_i))) \\
&\quad (\mathbf{pl}\text{ semi-commutes w/ cc-op}) \\
&= \overset{icc}{\sum} \Lambda_i u(\mathrm{conv}(\mathbf{pl}(w_i))) \\
&\quad (\mathrm{conv}\ \&\ u\text{ commute w/cc-op}),
\end{aligned}$$

where the first equality is due to the fact that utility functions and convex hull operator also commute, i.e. $\mathrm{conv}(u(K)) = u(\mathrm{conv}(K))$ for all $K \subseteq \mathcal{S}$. These equalities illustrate the idea of switching from probability numbers to probability intervals, in the context of projecting and evaluating abstract plans, by replacing the probability cross-product with the cc-operator. The interesting interplay among abstract actions/plans, utility functions, convex hull, and the cc-operator plays a key role in projecting and evaluating abstract probabilistic plans in the DRIPS planner. This important property allows the process of evaluating abstract plans to be performed in two steps. In the first step, we project each branch of each action in the plan on each branch of the current world, expanding a *pcc-tree* in the process (a world is now a set of probability distributions represented by a pcc-tree). In the second step, we compute the expected utility interval of the plan by replacing the leaves of the final pcc-tree with their utility values and evaluating the resulting *icc-tree.* The reader is refered to [13] for a more thorough discussion of various projection algorithms for DRIPS.

The next theorem states that the convex hull of projecting an abstract action/plan on a world can be com-

[3]We use the term "world" to describe the state of affairs after projecting several abstract actions. Since abstract actions are mappings that return *sets of probability distributions*, a world now can be a set of probability distributions.

puted by projecting on the convex hull of that world. This result is heavily utilized by DRIPS, since it is often convenient to approximate a set of probability distributions by its convex hull.

Theorem 4 *Suppose $w \subseteq S$, and $\mathbf{A}$ is an abstract action or an abstract plan. Then $conv(\mathbf{A}(w)) = conv(\mathbf{A}(conv(w)))$.*

Proof: The "$\subseteq$" relation is obvious, so we only need to prove the "$\supseteq$" relation, which amounts to proving that $\mathbf{A}(\text{conv}(w)) \subseteq \text{conv}(\mathbf{A}(w))$. An element of $\mathbf{A}(\text{conv}(w))$ must have the form $A(\overset{cc}{\sum}{}_{i=1}^{n} \lambda_i P_i)$, where $A \in \mathbf{A}$ and $P_i \in w$. Since concrete actions/plans commute with the cc-operator, this cc-sum can be written as $\overset{cc}{\sum}{}_{i=1}^{n} \lambda_i A(P_i)$, which is clearly an element of $\text{conv}(\mathbf{A}(w))$. □

5 INTERVAL BAYESIAN NETWORKS

In this section we demostrate how the cc-operator can be used to make inference in interval Bayesian networks.

A Bayesian Network is a pair consisting of a directed acyclic graph, and a collection of probability tables associated with the nodes. Each node of the graph represents a random variable, and its associated probability table specifies the conditional probability of that node given its parents. A Bayesian network represents a joint probability distribution over the random variables corresponding to the nodes, and this distribution is obtained by multiplying the conditional probability tables. Figure 5.a depicts a 4-node Bayesian network N, where the conditional probabilities to be specified are $P(X), P(Y|X), P(Z|X)$, and $P(T|Y,Z)$ (for example, $P(y|x) = .4$, and $P(\bar{y}|x) = .6$). This Bayesian network represents the joint probability distribution P over $\{X, Y, Z, T\}$: $P(X, Y, Z, T) = P(X)P(Y|X)P(Z|X)P(T|Y, Z))$.

Interval Bayesian networks (IBN) [10, 25, 8] are a generalization of Bayesian networks where we allow the probabilities to be interval-values. An IBN represents a set of Bayesian networks, each of which is obtained from the IBN by *instantiation*, a process that replaces each interval probability with a *consistent* point probability, where consistency means that the points must be in their corresponding intervals, and obey the axioms of probability. Figure 5.b depicts a 4-node interval Bayesian network $\mathbf{N}$ where $\mathbf{P}(y|x) = [.3\ .5]$ and $\mathbf{P}(\bar{y}|x) = [.5\ .7]$. The regular Bayesian network in Figure 5.a (N) is an instantiation of $\mathbf{N}$. We denote the

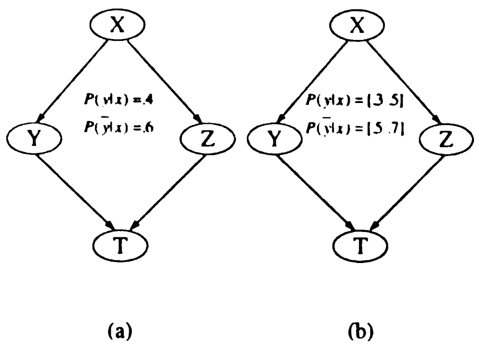


Figure 5: Examples: (a) a Bayesian network and (b) an interval Bayesian network.

instantiation relationship by $N \in \mathbf{N}$.

Given such an IBN $\mathbf{N}$, the answer to any probabilistic query is defined to be an interval whose endpoints are the infimum and supremum of that query, ranging over all Bayesian network instantiations of $\mathbf{N}$. For example, in the IBN in Figure 5.b, we might be interested in computing $\mathbf{P}(t)$, which is defined as $[\inf_{N \in \mathbf{N}} P_N(t)\ \sup_{N \in \mathbf{N}} P_N(t)]$.

Dechter [6] showed that most of Bayesian network algorithms for updating probabilities, finding the most probable explanation, finding the maximum a posteriori hypothesis, and computing maximum expected utility, can be cast into a unifying algorithm schema called *bucket elimination*. We now present an interval version of this schema and that computes the bounds for the probability of any node in an IBN. In this section we consider the case when there is no evidence in the IBN. The problem of answering probabilistic queries in IBNs with evidence will be studied in the Section 7 where we discuss evidential updating with pcc-trees.

The main idea of this algorithm can be best understood through an example. Suppose that in the IBN $\mathbf{N}$ in Figure 5.b, we are interested in computing the probability bounds for the random variable T taking on some value t. Suppose that P is the joint distribution represented by a BN instantiation N of $\mathbf{N}$. Note that $P(t)$ can be written as

$$\begin{aligned} P(t) &= \sum_{X,Y,Z} P(X, Y, Z, t) \\ &= \sum_{X,Y,Z} P(X)P(Y|X)P(Z|X)P(t|Y, Z). \end{aligned}$$

Our goal is to factorize this last sum into a series of probability cross-products, which can be generalized into a series of cc-operators. For example, we have that:

$$
\begin{aligned}
P(t) &= \sum_{X,Y,Z} P(X)P(Y|X)P(Z|X)P(t|Y,Z) \\
&= \sum_X P(X) \sum_Y P(Y|X) \sum_Z P(Z|X)P(t|Y,Z).
\end{aligned}
$$

This factorization corresponds exactly to what we get by applying the bucket elimination algorithm to the ordering $\{X, Y, Z, T\}$ of the random variables. From the fact that $P(X) \in \mathbf{P}(X), \sum_X P(X) = 1$, we can see that this formula can be generalized into a cc-operator sum. Now, given that X is fixed at some value x, we have that $P(Y|x) \in \mathbf{P}(Y|x), \sum_Y P(Y|x) = 1$, which means that we can generalized the inside sum into a cc-operator sum. It is obvious that all of the remaining sums in this formula can be generalized into cc-operator sums. Thus we have:

$$
\begin{aligned}
P(t) &= \sum_{X,Y,Z} P(X)P(Y|X)P(Z|X)P(t|Y,Z) \\
&= \sum_X P(X) \sum_Y P(Y|X) \sum_Z P(Z|X)P(t|Y,Z) \\
&\in \overset{icc}{\sum}{}_X \mathbf{P}(X) \overset{icc}{\sum}{}_Y \mathbf{P}(Y|X) \overset{icc}{\sum}{}_Z \mathbf{P}(Z|X)\mathbf{P}(t|Y,Z).
\end{aligned}
$$

The algorithm for the general case is the following. Suppose that we are interested in computing the bounds of the probability $\mathbf{P}(X)$ for some node X in an IBN $\mathbf{N}$. Let $\{X_1, X_2, \ldots, X_n\}$ be *any* topological (or ancestral) ordering of the random variables of $\mathbf{N}$, and $X = X_k$. Let $\pi_i, i = 1, \ldots, k$ denote the set of parents of node X_i. Then, the bounds of $\mathbf{P}(X_k = x_k)$ can be approximated as:

$$
\begin{aligned}
\mathbf{P}(x_k) &= \sum_{X_1,\ldots,X_{k-1}} \mathbf{P}(X_1, \ldots, X_{k-1}, x_k) \\
&= \sum_{X_1,\ldots,X_{k-1}} \prod_{i=1}^{k} \mathbf{P}(X_i|\pi_i) \\
&\subseteq \overset{icc}{\sum}{}_{X_1} \mathbf{P}(X_1) \overset{icc}{\sum}{}_{X_2} \mathbf{P}(X_2|\pi_2) \overset{icc}{\sum}{}_{X_3} \cdots \\
&\quad \overset{icc}{\sum}{}_{X_{k-1}} \mathbf{P}(X_{k-1}|\pi_{k-1})\mathbf{P}(x_k|\pi_k).
\end{aligned}
$$

Note that the bounds for $\mathbf{P}(x_k)$ obtained by this algorithm are correct, but not tight. The same can be said of the algorithm suggested in Theorem 3 for projecting and evaluating abstract plans. What's more, the two problems (of evaluating abstract plans and answering IBN quesries) look eerily similar, and a closer look at the two problems reveals that the latter is in fact a special case of the former: answering an IBN query can be viewed as an instance of abstract plan evaluation with suitable choices of worlds and actions. The precision loss of the algorithm for evaluating abstract plans is due to the fact that although the icc-operator sum – when applied only once – yields exact results, computations of globally related icc-operators are performed *locally* and *independently*. (An analogue of this loss occurs when we approximate the maximization of the sum of *correlated* individuals by summing the maximization of the individuals.) This is the precision tradeoff we have to make if we are to avoid having to resort to computationally more costly techniques such as linear programming. Experiments with the DRIPS planner show that the icc-tree approximation produces sufficiently tight bounds in order to eliminate large classes of suboptimal plans.

6 PCC-TREES AS A REPRESENTATION FOR SETS OF DISTRIBUTIONS

In this section, we investigate pcc-trees as data structures that encode sets of probability distributions. We first show that the class of pcc-trees is exactly the class of polytope-like sets of probability distributions, and discuss the complexity of switching between these two representations. We then show that Dempster-Shafer belief functions, when interpreted as lower envelopes of sets of distributions, can be represented by 2-level pcc-trees in at least two different ways.

6.1 PCC-TREES AND POLYTOPE-LIKE SETS OF DISTRIBUTIONS

One of the first question we need to answer when introducing pcc-trees is: "What kind of sets of probability distributions do pcc-trees encode?". It follows from Theorem 1 (the cc-operator and convex hull operator commute) that any pcc-tree is equal to its convex hull and thus encodes a convex set of probability distributions. Also recall that Figure 2.b demonstrates that any polytope-like set of probability distributions can be represented by a 2-level pcc-tree where the nodes at level 1 are the vertices of the polytope (this pcc-tree has a total of at most $1 + n + nd$ nodes, where n is the number of the vertices of the polytope, and $d = |\Omega|$). Our next result states that the converse is also true: the cc-operator is closed on the set of polytopes, and thus any pcc-tree represents a polytope-like set of probability distributions. In other words, the class of pcc-trees is precisely the class of polytope-like sets of probability distributions.

Theorem 5 *If $w_i \subseteq \Re^d$ are polytopes, then $K = \vec{\Lambda} \otimes \vec{w} = \overset{cc}{\sum}{}_{i=1}^{n} \Lambda_i w_i$ is also a polytope.*

Proof: We need the following auxiliary lemma, which is a standard result in convex geometry. (See, for example [11].)

Lemma 1 *Suppose that $K \subseteq \Re^d$ is a convex set and $V \subseteq K$ is a finite set of points in K such that for any point $u \in \Re^d$, there is a point $v \in V$ such that $\langle v, u\rangle = \max_{x \in K}\langle x, u\rangle$. Then $K = conv(V)$, or, equivalently, K is a polytope whose vertices are contained in V.*

Now consider any point $u \in \Re^d$. The set of real numbers that are inner products of u with the elements of K can be written as $\langle u, K\rangle = \langle u, \overset{cc}{\sum}{}_{i=1}^{n} \Lambda_i w_i\rangle = \overset{icc}{\sum}{}_{i=1}^{n} \Lambda_i \langle u, w_i\rangle$. (In the case when $K, w_i \subseteq \mathcal{S}$, we can view the set K as a set of probability distributions, u as a utility function, and $\langle u, K\rangle$ as the expected utility interval of an abstract plan whose final world is K.) It is easy to see that in order to compute the bounds of the interval $\langle u, w_i\rangle$, we only have to minimize and maximize $\langle u, x\rangle$ over the *finite* set of vertices of the polytope w_i. From this and the fact that the weights λ_i (chosen by the greedy knapsack algorithm for evaluating icc-operator sum, see Section 3) are chosen based only on the *order* of the values of the material (there are $n!$, i.e. *finite* number of possible orders), it follows that there are a finite set of points in K that satisfies the condition of the auxiliary Lemma. □

It is not an easy task to determine the vertices of a polytope represented by a pcc-tree. It is however possible to estimate the number of vertices of a polytope represented as a pcc-tree. We have found that the number of vertices of a polytope represented as a pcc-tree with n nodes is asymptotically bounded above by $O((2e^{1/e})^n)$. The proof of this result is rather involved and is omitted here. We can obtain a very rough upper bound using the following argument. Denote by $v(K)$ the number of vertices of a polytope K. Denote by $\mathcal{PCC}(n)$ the set of polytopes representable by pcc-trees of at most n nodes, and $v(n) = \sup_{K \in \mathcal{PCC}(n)} v(K)$. Now assume that $K = \overset{cc}{\sum}{}_{i=1}^{m} \Lambda_i w_i, 2 \leq m \leq n$ is a polytope represented by a pcc-tree with $(n+1)$ nodes, where w_i are also polytopes. Since w_i are polytopes representable with pcc-trees of at most n (in fact $n-1$, since $m \geq 2$) nodes, we have that $v(w_i) \leq v(n)$. The proof of Theorem 5 implies that $v(K) \leq m!(v(n))^m \leq n!(v(n))^n$. Since K was chosen arbitrarily, we have that $v(n+1) \leq n!(v(n))^n$.

Although the recursive upper bound obtained above is clearly very loose, it has an implication that there are polytopes (with sufficiently large number of vertices) that require pcc-trees with arbitrarily large number of nodes to represent them. Another implication of this bound is the following theorem.

Theorem 6 *For all positive integers n, $\mathcal{PCC}(n)$ is a strict subset of $\mathcal{PCC}(n+2)$.*

Proof: Let $K \in \mathcal{PCC}(n)$ and $v(K) = v(n)$. Such K exists since $v(n) < \infty$. We will construct from K a polytope K' representable by a pcc-tree with $n+2$ nodes which does not belong to $\mathcal{PCC}(n)$.

Fix a point $s \in \mathcal{S} - K$ (such a point must exist unless K is the entire probability simplex). Let $I = [0\ 1/2]$, $J = [1/2\ 1]$, and $K' = (I, J) \otimes (\{s\}, K)$. Also set $K(s, 1/2) = \{1/2(s+x) | x \in K\}$. It is easy to verify that 1) $K' = \text{conv}(K(s,1/2) \cup K)$ and 2) $v(\text{conv}(K(s,1/2) \cup K)) > v(K)$, which establish the theorem. □

6.2 PCC-TREES AND DEMPSTER-SHAFER BELIEF FUNCTIONS

In this subsection we show that any Dempster-Shafer belief function [7, 21], when viewed as a lower envelope of a set of probability distributions, can be represented by a 2-level pcc-tree, using its corresponding mass assignment function. We first need some preliminaries to formulate this result.

Let Ω be a finite set [4]. A function $Bel : 2^\Omega \to [0,1]$ is called a *belief function* if $Bel(\emptyset) = 0$, $Bel(\Omega) = 1$, $Bel(A) \leq Bel(B)$ whenever $A \subseteq B$, and for all $A_i \subseteq \Omega, i = 1, \ldots, k$:

$$Bel\left(\bigcup_{i=1}^{k} A_i\right) \geq \sum_{\emptyset \neq I \subseteq \{1,\ldots,k\}} (-1)^{|I|+1} Bel\left(\bigcap_{i \in I} A_i\right).$$

The corresponding *mass assignment function* [5] $m : 2^\Omega \to [0,1]$ of a belief function Bel is obtained by the Möbius transformation as $m(A) = \sum_{B \subseteq A} (-1)^{|A-B|} Bel(B), \forall A \subseteq \Omega$. It is well-known that $\sum_{A \subseteq \Omega} m(A) = 1$. The subsets $A \subseteq \Omega$ that are assigned positive masses ($m(A) > 0$) are called the *focal*

[4] Ω is often called *frame of discernment.*

[5] Also called *basic probability assignment.*

elements of the belief function Bel. A belief function Bel can also be viewed as the lower envelope of a set of probability distributions *consistent* with it, where a probability distribution P is consistent with Bel if $Bel(A) \leq \sum_{s \in A} P(s), \forall A \subseteq \Omega$.

Kyburg [17] showed that the set of probability distributions consistent with a belief function is closed and convex. Our next theorem [13] asserts something stronger: this set is a polytope-like set of probability distributions, which is equal to the weighted sum of the faces of the probability simplex, with faces $\mathrm{conv}(A), A \subseteq \Omega$ weighted with $m(A)$. A proof of this theorem can be found in [12].

Theorem 7 *$Bel = \overset{cc}{\sum}{}_{i=1}^{k} m(A_i) conv(A_i)$. Here both sides are interpreted as sets of probability distributions (See Figure 2.c for an example).*

7 EVIDENTIAL UPDATING WITH PCC-TREES

Theorem 5 raises a natural question: If the class of pcc-trees is identical to the class of polytope-like sets of probability distributions, why shoud not we just use well-known methods of representing polytopes such as vertex-listing, or linear programming? Given a pcc-tree, we can in principle either determine the vertices of the polytope it represents, or establish a system of linear inequalities that characterizes this polytope. In the abstraction-based probabilistic planning framework described in Section 4, neither of these two approaches appears to be very efficient. For example, if we represent uncertainty of the world by the list of vertices of a polytope, we need to compute this list after each action projection, which is a non-trivial task. Similarly, it seems no less difficult to determine a system of linear inequalities that characterizes uncertainty of the world after each action projection. Using pcc-trees, the plan projection process reduces to a simple tree-spanning process, and the plan evaluation (expected utility computation) process reduces to a simple icc-tree evaluation process. Furthermore, the cc-operator and pcc-trees have a number of properties that are particularly useful for the purpose of abstracting actions and plans ([13], Lemma 1-4).

However, in the context of evidential updating, the pcc-tree representation turns out to be less suitable than its vertex-listing and linear programming counterparts. We investigate the reasons of this limitation of pcc-trees in this section.

The Bayesian approach to dealing with uncertainty involves representing the uncertain information with a *prior* probability distribution $P(.)$, and updating it with an observed evidence E to obtain a *posterior* probability distribution $P(.|E)$, which is defined as $P(x|E) = P(x, E)/P(E)$. Following [9], we view an evidence E as a function, called *evidential function* that maps probability functions to probability functions: $E : P \mapsto E(P) = P(.|E)$. For probability functions P where $P(E) = 0$, $E(P)$ is undefined. In this paper we consider only *certain* (also called *hard*) evidence E that is a subset of the sample space: $E \subseteq \Omega$.

The Bayesian paradigm is often challenged due to the its requirement that a *single* prior probability distribution be specified, even when little statistical information is available. The *cautious Bayesian* approach addresses this issue by taking into consideration a (usually convex) set K of prior probability distributions, and updating each member $P \in K$ of this set with the observed evidence E to obtain $E(K) = \{E(P) | P \in K, P(E) > 0\}$. This evidential updating approach is refered to as *convex conditionalization* [19, 17].

Convex conditionalization for various representations of convex sets of probability distributions has been a subject of extensive study. White [28] studied the case where both the prior set of distributions and the (uncertain, also called *soft*) evidence are characterized by systems of linear inequalities. Snow [23] showed that in the case where the evidence is a single uniformly positive distribution, White's method of conditioning yields exact results.

Convex conditionalization has also been studied in the case where the set K of prior distributions is a set of distributions consistent with a Dempster-Shafer belief function Bel. Fagin and Halpern [9] showed that the lower and upper envelopes of $E(Bel)$ correspond to a pair of belief (denoted $Bel(.|E)$) and plausibility functions. Jaffray [16] showed that $E(Bel)$ in general is a *strict* subset of the set of distributions consistent with the new belief function $Bel(.|E)$. This result means that the belief function representation loses information in the updating process. Chrisman [3] showed that this loss of information is especially troublesome when pieces of evidence are to be updated incrementally, since different orders of pieces of evidence produce different results (which are also different from the result obtained when updating all the pieces of evidence in a single step). By using Dempster conditioning bounds in conjunct with the convex conditioning bounds, Chrisman was able to develope an exact conditioning procedure for 2-monotone lower probabilities, the class of which contains the class of belief functions [3].

We now investigate the problem of convex conditioning on pcc-trees. Suppose that $K = \overset{cc}{\sum}{}_{i=1}^{n} \Lambda_i w_i$ is a pcc-trees where w_i are also pcc-trees that are subtrees of K. Suppose that $E \subseteq \Omega$ is hard evidence. We are interested in determining $E(K) = E(\overset{cc}{\sum}{}_{i=1}^{n} \Lambda_i w_i)$. We start out by studying this problem in special cases of pcc-trees.

In the case when $K = \text{conv}(A), A \subseteq \Omega$ is a face of the probability simplex (Figure 2.a), then clearly $E(K) = E(\text{conv}(A)) = \text{conv}(E \cap A)$. In the more general case when $K = \text{conv}\{P_1, \ldots, P_k\}$ is a polytope-like sets of probability distributions with vertices P_i, it can be shown that $E(K)$ is also a polytope whose vertices are contained in the set $\{E(P_i)|i = 1, \ldots, k\}$ [22, 24]. Below we provide a proof of this statement that is similar to the proof given in [24], since it is highly instructive in our upcoming analysis.

Theorem 8

$$E(conv\{P_1, \ldots, P_k\}) = conv(E(P_1), \ldots, E(P_k)\}).$$

Proof: Like most arguments involving convex combinations, here we need to consider only the case of $k = 2$. The case when either $P_1(E) = 0$ or $P_2(E) = 0$ is easy to check, so we suppose that $P_1(E) > 0, P_2(E) > 0$. An element of the left-hand side has the form $E(\alpha P_1 + (1 - \alpha)P_2), \alpha \in [0\ 1]$, while an element of the right-hand side has the form $\beta E(P_1) + (1 - \beta)E(P_2), \beta \in [0\ 1]$. The set equality is established if we can construct a one-to-one mapping from any $\alpha \in [0\ 1]$ to a $\beta \in [0\ 1]$ such that $E(\alpha P_1 + (1 - \alpha)P_2) = \beta E(P_1) + (1 - \beta)E(P_2)$. It is easy to check that the mapping $\beta = \alpha/[\alpha + (1 - \alpha)P_2(E)/P_1(E)]$ satisfies these requirements. □

An immediate observation from the above proof of Theorem 8 is that the evidential function E does not preserve convex combinations and thus is *not* a convex mapping [6]. As one would suspect, this non-convexity of the evidential function causes difficulty in updating pcc-trees, since the evidential function does not commute with the cc-operator.

An example that illustrates this point is the case of updating belief functions. Recall that a belief function Bel with its corresponding mass assignment function m and focal elements $A_1, \ldots, A_k$ can be written as $Bel = \overset{cc}{\sum}{}_{i=1}^{k} m(A_i)\text{conv}(A_i)$ (see Theorem 7 and Figure 2.c). If the evidential function E had commuted with the cc-operator, we would have obtained

[6]Further investigation based on geometric arguements shows that this phenomenon occurs because of the *normalization step* of updating, when the probabilities $P(x, E)$ are normalized by dividing by $P(E)$ so that they sum up to 1.

$$\begin{aligned} E(Bel) &= E(\overset{cc}{\sum}{}_{i=1}^{k} m(A_i)\text{conv}(A_i)) \\ &= \overset{cc}{\sum}{}_{i=1}^{k} m(A_i)E(\text{conv}(A_i)) \\ &= \overset{cc}{\sum}{}_{i=1}^{k} m(A_i)\text{conv}(A_i \cap E), \end{aligned}$$

which is identical to $Bel \oplus E$, the result of combining the belief function Bel with the evidence E using Dempster's rule of combination. This is impossible since it is well-known that the bounds obtained using Dempster's rule in general are *strictly narrower* than the bounds obtained using convex conditionalization [7, 9].

The difficulty of evidential updating with pcc-trees translates into the difficulty of answering probabilistic queries in interval Bayesian networks that have evidence. A straightforward attempt to extend the algorithm described in Section 5 to IBNs with evidence involves maintaining two different representations of polytope-like sets of probability distributions: with a pcc-tree and with a list of the polytope's vertices. We can use the second representation to incorporate evidence according to Theorem 8, and use the first representation to compute the bounds for probabilistic queries. Effective realization of this idea requires efficient algorithms for alternating between the two representations. We leave this as an issue for future research.

Acknowledgements

We would like to thank AnHai Doan for many useful discussions, Van Vu for proving Theorem 6, and the anonymous referees for their insightful comments and suggestions. This work was supported in part by NSF grant IRI-95-09165.

References

[1] C. Boutilier and R. Dearden. Using abstractions for decision-theoretic planning with time constraints. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 1016–1022, Seattle, July 1994.

[2] L. Chrisman. Abstract probabilistic modeling of action. In *Proceedings of the First International Conference on Artificial Intelligence Planning Systems*, pages 28–36, June 1992.

[3] L. Chrisman. Incremental conditioning of lower and upper probabilities. *International Journal of Approximate Reasoning*, 13(1):1–25, 1995.

[4] F. Cozman. Robustness analysis of bayesian networks with local convex sets of distributions. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 108–115, August 1997.

[5] T. Dean, L. Pack Kaelbling, J. Kirman, and A. Nicholson. Planning under time constraints in stochastic domains. *Artificial Intelligence*, 76(1-2):35–74, July 1995.

[6] R. Dechter. Bucket elimination: A unifying framework for probabilistic inference algorithms. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pages 211–219, August 1996.

[7] A. Dempster. Upper and lower probabilities induced by a multi-valued mapping. *Annals of Mathematical Statistics*, 38(2):325–339, 1967.

[8] D. Draper. *Localized Partial Evaluation of Bayesian Belief Networks*. PhD thesis, Computer Science Dept., Univ. of Washington, Seattle, November 1995.

[9] R. Fagin and J. Halpern. A new approach to updating belief. In P. Bonissone, M. Henrion, L. Kanal, and J. Lemmer, editors, *Uncertainty in Artificial Intelligence 6*, pages 347–374. 1991.

[10] K.W. Fertig and J.S. Breese. Probability intervals over influence diagrams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(3):280–286, March 1993.

[11] B. Grunbaum. *Convex Polytopes*. Interscience, London, New York, 1967.

[12] V. Ha, A. Doan, V. Vu, and P. Haddawy. Geometric foundations for interval-based probabilities. 1998. Submitted to the Annals of Mathematics and Artificial Intelligence 5.

[13] V. Ha and P. Haddawy. Theoretical foundations for abstraction-based probabilistic planning. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pages 291–298, Portland, August 1996.

[14] P. Haddawy, A. Doan, and C.E. Kahn Jr. Decision-theoretic refinement planning in medical decision making: Management of acute deep venous thrombosis. *Medical Decision Making*, 16(4):315–325, Oct/Dec 1996.

[15] S. Hanks. Practical temporal projection. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 158–163, Boston, July 1990.

[16] J. Jaffray. Bayesian updating and belief functions. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(5):1144–1152, 1992.

[17] H. Kyburg Jr. Bayesian and non-bayesian evidential updating. *Artificial Intelligence*, 31(3):271–294, 1987.

[18] H. Kyburg and M. Pittarelli. Set-based bayesianism. *IEEE Transactions on Systems, Man, and Cybernetics*, 26:324–339, 1996.

[19] I. Levi. *The Enterprise of Knowledge*. MIT Press, Cambridge,MA, 1980.

[20] T. Seidenfeld, M. Schervish, and J. B. Kadane. On the shared preferences of two bayesian decision makers. *Journal of Philosophy*, 86:225–244, 1989.

[21] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, Princeton, NJ, 1976.

[22] P. Snow. Bayesian inference without point estimates. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 233–237, 1986.

[23] P. Snow. Improved posteriori probability estimates from prior and conditional linear constraint systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(2):464–469, 1991.

[24] W. Stirling and A. Morrell. Convex bayesiandecision theory. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(1):173–183, 1991.

[25] B. Tessem. Interval probability propagation. *International Journal of Approximate Reasoning*, 7:95–120, 1992.

[26] S. Thiebaux, J. Hertzberg, W. Shoaff, and M. Schneider. A stochastic model of actions and plans for anytime planning under uncertainty. *International Journal of Intelligent Systems*, 1994.

[27] P. Walley. *Statistical Inference with Imprecise Probabilities*. Chapman and Hall, 1991.

[28] C. White. A posteriori re[resentations based on linear inequality descriptions of a priori and conditional probabilities. *IEEE Transactions on Systems, Man, and Cybernetics*, 16:570–573, 1986.

Making Decision in a Qualitative Setting: from Decision under Uncertainty to Case-based Decision

Didier Dubois
IRIT CNRS
Université Paul Sabatier,
31062 Toulouse France

Lluís Godo
IIIA CSIC,
Campus UAB
08193 Bellaterra, Spain

Henri Prade
IRIT CNRS
Université Paul Sabatier,
31062 Toulouse France

Adriana Zapico
IIIA CSIC,
Campus UAB
08193 Bellaterra, Spain

Abstract

The paper improves a previously proposed axiomatic setting for qualitative decision under uncertainty in the von Neumann and Morgenstern' style, where only ordinal scales are required for assessing uncertainty and utility. Two qualitative criteria are axiomatized in a finite setting: a pessimistic one and an optimistic one, respectively obeying an uncertainty aversion axiom and an uncertainty-attraction axiom. These criteria generalize the well-known maximin and maximax criteria, making them more realistic. They are suited to one-shot decisions. They are not based on the notion of mean value, but take the form of medians. Elements for a qualitative case-based decision methodology are also proposed, with pessimistic and optimistic evaluations formally similar to the expressions which cope with uncertainty, up to modifying factors which cope with the lack of normalization of similarity evaluations.

1 INTRODUCTION

In decision problems, given a (finite) set of situations S and a (finite) set of consequences X, a *decision* d is represented by a function d: S → X, often called an *act*, and the preference relation among consequences is modelled by a utility function u: X → U, where U is a preference scale, usually a numerical scale. Uncertainty in decision problems may appear in different stages. For instance, it may occur that:

- the acts are precisely defined, but we do not know what the real situation is,
- the real situation is known, but the consequences of decisions are not precisely known, or even partially specified.

Classical approaches to decision making under uncertainty assume that uncertainty is represented by probability distributions. For instance, if a probability distribution P on the set of situations (also called *probabilistic lottery*) is assumed to be known, and a numerical utility $u: X \rightarrow \mathbb{R}$ is assigned to consequences, then decisions $d: S \rightarrow X$ can be ranked according to the expected value of the random variable $u \circ d$ with respect to P. Namely, decision d' is at least as good as d, denoted $d \sqsubseteq d'$, iff $E(d) \leq E(d')$, where the utility function E is defined as $E(d) = \sum_{s \in S} P(s) \cdot u(d(s))$, and it is proved to preserve the *natural* combination operation of probabilistic lotteries defined by convex linear combination. This approach corresponds to the well-known Expected Utility Theory (EUT) for which Von Neumann and Morgenstern [20] and Savage [23] provided nice axiomatic frameworks .

One of the problems of EUT is that it needs numerical probabilities for each state and numerical utilities for all possible consequences. Sometimes this assumption is too strong if there is only incomplete or poor available information. In these cases a more qualitative approach is needed. Moreover EUT is especially tailored for repeated decisions whose results accumulate additively. This is the underlying meaning of the averaging nature of expected utility. However in the case of one-shot decisions or decisions whose individual results do not compensate each other, EUT does not yield a convincing criterion for rank-ordering decisions. This situation of non-additivity naturally occurs with qualitative information about the worth of consequences.

The classical axiomatic frameworks of utility theory have actually been questioned rather early, challenging some of the postulates leading to the expected utility criterion. Noticeably, Allais [1] and later Ellsberg [14] laid bare the existence of cases where a systematic violation of the expected utility criterion could be observed. Some of these violations were due to a cautious attitude of decision-makers. More recently Gilboa [15] and Schmeidler [24] have advocated and axiomatized lower and upper expectations expressed by Choquet integrals attached to non-additive numerical set-functions (corresponding to a family of probability measures) as a formal approach to utility that accounts for Ellsberg paradox (see also [22]). One of these generalized expected utility criteria (the lower expectation) is also a numerical generalization of the cautious Wald criterion for decision under ignorance. The latter suggests that a decision be evaluated by the value of its worst possible consequence. Choquet integrals, especially the lower expectations, are mild versions of Wald criterion. The pessimistic (resp. optimistic) criterion proposed here can again be viewed as a refinement of Wald criterion (resp. the maximax criterion), but the utility functions are qualitative, hence reject the notion of averaging put forward by the classical theory, and also sanctioned by Choquet integrals.

Steps to qualitative decision theory have been investigated in various directions by AI researchers recently. Some approaches are based on an all-or-nothing notions of utility and/or plausibility (e.g., Bonet and Geffner [4], Brafman and Tennenholtz [5]). The latter clearly advocates Wald cautious criterion. Others use integer-valued functions (e.g., Pearl [21], Tan and Pearl [27]). Boutilier [3] suggests to focus on the states with maximum plausibility only, a policy examined by Dubois et al. [8] and that leads to debatable decisions.

Our approach, already outlined by Dubois and Prade in [11], is based on an axiomatic framework that is a *qualitative* counterpart to Von Neumann and Morgenstern's Expected Utility Theory. In this approach, both preferences on consequences and uncertainty are graded on ordinal scales (i.e., scales equipped with the maximum, minimum and an order-reversing operation) that are commensurate. This qualitative decision theory appears as the natural decision theory related to Possibility Theory. This paper improves the previous axiomatics and representation results of [11]. Moreover, we propose two axiomatic frameworks that enable preferences on possibilistic "lotteries" to be represented by pessimistic and optimistic qualitative utilities respectively, defined, one as the *necessity* and the other as the *possibility* of a fuzzy event [32,10], a particular type of the so-called Sugeno integrals [26].

After this Introduction, in the following section we describe the possibilistic framework for representing uncertainty and preference. We introduce the proposed qualitative utilities and explain their meaning on the basis of an example. Section 3 summarises Dubois and Prade's proposal and compares it to the standard Von Neumann Morgenstern utility framework. In Sections 4 and 5 we present our improved axiomatics with their corresponding representation theorems for pessimistic and optimistic Decision Maker's behaviours. Section 6 discusses the application of the decision model for case-based decision.

2 POSSIBILITY THEORY AND QUALITATIVE UTILITY

A possibility distribution π on a (finite) set of possible worlds or states S is a mapping from S to a bounded, linearly ordered valuation set (V, <). This ordered set can be anything, but it may be exemplified by the unit interval [0,1], equipped by the maximum and the minimum operation, and an order-reversing map that is denoted n_V, such that $n_V(0) = 1$ and $n_V(1) = 0$. So, a possibility distribution is not necessarily numerical. In the numerical setting, when V = [0, 1], function n_V is generally taken as $1 - (\cdot)$. When V is finite, we continue to denote 0 and 1 the top and bottom of V. The referential set S represents a set of "states of affairs"or possible worlds, each state being an unambiguous description of a cluster of situations, at a certain level of granularity. Possibility is often understood in terms of plausibility. However possibility distributions can represent preference as well and then possibility means "feasibility".

2.1 Qualitative possibility theory

A possibility distribution describes knowledge about the unknown value taken by one or several attributes used to describe states of affairs. For instance it may refer to the age of a man, the size of a building, the temperature of a room, etc. Here it will refer to the ill-known consequence of a decision. A possibility distribution represents a state *of knowledge* (about the state of affairs) distinguishing what is plausible from what is less plausible, what is the normal course of things from what is not, what is surprizing from what is expected. The function $\pi: S \to V$ represents a flexible restriction on the actual state of affairs, with the following conventions:

$\pi(s) = 0$ means that state s is rejected as impossible;
$\pi(s) = 1$ means that s is totally possible (= plausible).

Distinct states may simultaneously have a degree of possibility equal to 1. Flexibility in this description is modelled by letting $\pi(s)$ between 0 and 1 for some states s. The quantity $\pi(s)$ thus represents the degree of possibility of the state s, some states being more possible than others. Clearly, if S is the complete range of states, at least one of the elements of S should be fully possible, so that $\exists s, \pi(s) = 1$ (normalization). In this paper we consider mostly normalized possibility distributions (except in the last section). Strictly speaking a possibility distribution can be viewed as the generalized characteristic function of a fuzzy set [31]. The fundamental point made by Zadeh [32] is the following: as set-characteristic functions can be used to express equipossibility, fuzzy set membership functions are the basis of gradual possibility.

A possibility distribution π is said to be *at least as specific* as another π' if and only if for each state of affairs s: $\pi(s) \le \pi'(s)$ (Yager [30]). Then, π is at least as restrictive and informative as π'. In the possibilistic framework extreme forms of partial knowledge can be captured, namely

- complete knowledge: for some s_0, $\pi(s_0) = 1$ and $\pi(s) = 0, \forall s \neq s_0$ (only state s_0 is possible)
- complete ignorance: $\pi(s) = 1, \forall s$ (all states in S are possible).

To each possibility distribution π, we can associate its comparative counterpart, a complete preorder denoted by $\ge_\pi$, defined by $s \ge_\pi s'$ if and only if $\pi(s) \ge \pi(s')$, which induces a well-ordered partition [25] $\{E_1, \ldots, E_n\}$ of S, that is, $\{E_1, \ldots, E_n\}$ is a partition of S such that $\forall s \in E_i, \forall s' \in E_j, \pi(s) \ge \pi(s')$ iff $i \le j$ (for $1 \le i, j \le n$).

By convention E_1 represents the most normal states of fact. Thus, a possibility distribution partitions S into classes of equally possible states.

In the following, subsets are denoted A, B, C, ... $\overline{A}$ denotes the complement of A. Given a simple query of the form "does the actual state belong to A?", where A is a prescribed subset of situations, the response to the query can be obtained by computing the partial belief induced on A by the knowledge encoded by the possibility distributon π, noticeably to what extent:

- A is consistent with π, with degree
  $\Pi(A) = \sup_{s \in A} \pi(s)$,
- A is certainly implied by π, with degree
  $N(A) = n_V(\Pi(\overline{A})) = \inf_{s \notin A} n_V(\pi(s))$.

$\Pi(A)$ is called the degree of possibility of A, and defined by assuming that, if it is only known that A occurs, then the most plausible situation compatible with A is the one that takes place. It expresses a level of unsurprizingness. The basic axiom of possibility measures is $\Pi(A \cup B) = \max(\Pi(A), \Pi(B))$. It is justified by the assumption of jumping to the most plausible situation. By convention, $\Pi(\emptyset) = 0$.

N(A) is called degree of necessity of A. $N(A) \geq \alpha > 0$ means that the most plausible situation where A is false is rather impossible, i.e., not possible to a level greater than $n_V(\alpha)$. Necessity measures satisfy an axiom dual of the one of possibility measures, namely $N(A \cap B) = \min(N(A), N(B))$. This decomposability axiom, as well as the above maxitivity axiom, presupposes a finite setting in order to be characteristic. Otherwise the axiom must hold for infinite families of sets.

Set-functions Π and N are respectively called *possibility* and *necessity measures* [10], and can provide simple ordinal representations of graded belief that are fully compatible with preferential representations of uncertainty very common in nonmonotonic reasoning [2]. Their particular character lies in their ordinal nature, i.e., the valuation set V is used only to rank-order the various possible situations in S, in terms of their compatibility with the normal course of things as encoded by the possibility distribution π. As said above, a systematic assumption in possibility theory is that the actual situation is normal, i.e., it is any s such that $\pi(s)$ is maximal given other known constraints. It justifies the evaluation $\Pi(A)$, and contrasts with the probabilistic evaluation of the likelihood of events. Moreover $N(A) > 0$ means that A holds in all the most normal situations. Since the assumption of normality is always made, $N(A) > 0$ thus means that A is an accepted belief, i.e., one may act as if A were true. This assumption is always a default one and can be revised if further pieces of evidence contradict it.

2.2 Possibilistic utility

Let us stress two distinct understandings of a possibility distribution. A possibility distribution may encode imprecise knowledge about a situation, as in the previous section. In that case, no choice is at stake, that is, the actual situation is what it is, and π encodes plausible guesses about it. However, possibility distributions can also express which are the states in which an agent would like to be, under the form of a flexible constraint on the state space. In this case possibility is interpreted in terms of graded preference or subjective feasibility. A possibility distribution is then similar to a utility function, or, better, a value function, but it may range on a qualitative valuation set (see also [7] for a detailed discussion of the preference view of possibility theory in the setting of constraint satisfaction).

In the general case, decisions are made in an uncertain environment. In the Savage framework, the consequence of a decision depends on the state of the world in which it takes place. If S is a set of states and X a set of possible consequences, the decision-maker has some knowledge of the actual state and some preference on the consequences of his decision. It makes sense, if information is qualitative, to represent the incomplete knowledge on the state by a possibility distribution π on S with values in a plausibility scale V and the decision-maker's preference on X by means of another possibility distribution u with values on a (ordinal) preference scale U. The utility of a decision d whose consequence in each state s is $x = d(s) \in X$, can be evaluated by combining the plausibilities $\pi(s)$ and the utilities u(x) in a suitable way.

Notice that if we have a possibility distribution π on the set of situations S, then every decision $d: S \rightarrow X$ induces a corresponding possibility distribution π_d on the set of consequences defined as $\pi_d(x) = \max\{\pi(s) \mid d(s) = x\}$, assuming $\max \emptyset = 0$. So, in this framework, to rank decisions amounts to ranking possibility distributions in the set of normal possibility distributions on X over V, denoted Pi(X). Therefore, we shall be interested in preference relations and utility functions defined on Pi(X) rather than in S.

In this framework, Dubois and Prade [11] propose to use two kinds of qualitative utility functions to rank order possibility distributions (or possibilistic lotteries, if you like). The basic underlying idea is the following one. A qualitative utility function $u: X \rightarrow U$ on the consequences can be regarded as specifying a preference profile: the greater is u(x), the more preferred is the consequence x. On the other hand a possibility distribution $\pi: X \rightarrow V$ specifies which consequences are plausible: the greater $\pi(x)$, the more plausible is the consequence x. Therefore, a pessimistic (or conservative) criterion is to look for those π's which, to some extent, make all the bad consequences hardly plausible. On the contrary, an optimistic criterion is to look for those π's that, also to some extent, make at least one of the good consequences highly plausible.

However, to define such criteria an assumption of *commensurateness* between the plausibility scale V and the preference scale U has to be made. For the moment, what we basically need is an order preserving mapping $h: V \rightarrow U$ such that $h(1) = 1$ and $h(0) = 0$. Then the following pessimistic and optimistic qualitative utilities can be defined respectively:

$$QU^-(\pi) = \min_{x\in X} \max(n_U(\pi^*(x)), u(x)) \quad (1)$$

$$QU^+(\pi) = \max_{x\in X} \min(\pi^*(x), u(x)) \quad (2)$$

where $\pi^*(x) = h(\pi(x))$ and n_U: $U \rightarrow U$ is the order reversing involution on U. The optimistic criterion has been first proposed by Yager [29] and the pessimistic criterion by Whalen [28]. One can easily notice that $QU^-(\pi)$ and $QU^+(\pi)$ are nothing but the *necessity* and *possibility* measures of the fuzzy set F, with U-valued membership function $\mu_F = u$, with respect to the distribution π^* [10,32], namely $QU^-(\pi) = N_{\pi^*}(F)$ and $QU^+(\pi) = \Pi_{\pi^*}(F)$. They respectively account for a degree of inclusionhip (of π^* into F) and a degree of intersection. When the uncertainty representation π^* is all or nothing ($\pi^*(X) = \{0, 1\}$), QU^- and QU^+ turn out to yield the well-known *maximin* and *maximax* decision criteria, respectively. QU^- generalizes the max-min Wald criterion in the absence of probabilistic knowledge since if π^* is the characteristic function of a crisp set A:

$$QU^-(\pi) = N_{\pi^*}(F) = \min_{x\in A} u(x),$$

that is, the worst consequence compatible with π^* is used to assess the utility of the decision underlying π^*. Similarly, the optimistic criterion would come down to computing $QU^+(\pi) = \max_{x\in A} u(x)$ (criteria used in [5]). Our extensions of these criteria neglect the implausible consequences of the decision, and are either pessimistic and optimistic on the remaining consequences, the threshold between plausibility and implausibility depending on the decision and on the commensurateness mapping.

It is interesting to consider the form of the qualitative utilities on simple possibility distributions on X called *qualitative binary lotteries* $\pi = (\lambda/x, \mu/y)$ defined as follows,

$$(\lambda/x, \mu/y)(x) = \lambda,$$
$$(\lambda/x, \mu/y)(y) = \mu,$$
$$(\lambda/x, \mu/y)(z) = 0 \text{ for } z \neq x, z \neq y,$$

where $x, y \in X$, $\lambda, \mu \in V$ such that $\max(\lambda, \mu) = 1$. They correspond to a binary act which produces x if some event occurs, and y otherwise. Assume $u(x) > u(y)$. Then it is easy to check that, denoting $n = n_U \circ h$,

$$QU^-(\pi) = \min(\max(n(\lambda), u(x)), \max(n(\mu), u(y)))$$
$$= \min(u(x), \max(n(\mu), u(y)))$$

Indeed if $\lambda = 1$ then the result is obvious. If not then $\mu = 1$ and $QU^-(\pi) = \min(\max(n(\lambda), u(x)), u(y)) = u(y) = \min(u(x), \max(n(\mu), u(y)))$ since $u(x) > u(y)$. This form of the pessimistic utility is easy to understand: if the agent is sure enough that x obtains ($N(\{x\}) = n(\mu) > u(x)$) then the utility of the decision that gives x or y is u(x). If the agent has too little knowledge ($\max(N(\{x\}), N(\{y\}) < u(y))$ he is cautious and the utility is u(y), the worst case. Of course the same happens if the agent is at least somewhat certain that y obtains. If the agent's certainty that x obtains is positive but not extreme, the utility reflects the certainty level and is equal to $n(\mu)$. Note that the utility of the binary qualitative lottery is the median of $\{u(x), n(\mu), u(y)\}$, thus contrasting with expected utility, which is a mean.

Similarly, the optimistic utility of the binary act takes the simplified form:

$$QU^+(\pi) = \max(\min(h(\lambda), u(x)), u(y))$$

and can be interpreted similarly, but here the utility is u(x) as soon as the agent believes that obtaining x is possible enough ($\Pi(\{x\}) = h(\lambda) > u(x)$).

Example: Consider the omelette example of Savage ([23], 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 (BIO); break it apart in a cup (BAC); or throw it away (TA). Assume that utilities and degrees of necessity belong to the same totally ordered scale, here {0, a, b, c, d, 1}, where $0 < a < b < c < d < 1$, equipped with its order-reversing map n. The set of 6 consequences is given in Table 1. Grades between parentheses indicate an arbitrary encoding of the utility ordering of consequences. The reader can easily check that he/she agrees with this ordering. Only two states (Fresh F, Rotten R) are present, so that we deal with binary acts. The pessimistic utilities of the three acts in the egg example are then given as a function of states F and R:

Table 1: States, acts and consequences in Savage's omelette example

STATES / ACTS	fresh egg (F)	rotten egg (R)
break the egg in the omelette (BIO)	a 6 egg omelette (1)	nothing to eat (0)
break it apart in a cup (BAC)	a 6 egg omelette, a cup to wash (d)	a 5 egg omelette, a cup to wash (b)
throw it away (TA)	a 5 egg omelette, one spoiled egg (a)	a 5 egg omelette (c)

Table 2: Qualitative utilities in Savage's omelette example

N(F)	N(R)	QU⁻(BIO)	QU⁻(BAC)	QU⁻(TA)	Best acts
1	0	1	d	a	BIO
d, c, b	0	N(F)	N(F)	a	BIO or BAC
a	0	a	b	a	BAC
0	0, a	0	b	a	BAC
0	b	0	b	b	BAC or TA
0	≥ c	0	b	c	TA

$$QU^-(BIO) = \min(1, \max(N(F), 0)) = N(F)$$
$$QU^-(BAC) = \min(d, \max(N(F), b))$$
$$QU^-(TA) = \min(\max(N(R), a), c)$$

where $N(F)=1-\Pi(R)$, $N(R)=1-\Pi(F)$ and $\min(N(F), N(R)) = 0$. Table 2 exhibits the best acts as a function of the decision-maker's knowledge about the egg. The model recommends act BAC in case of relative ignorance on the egg state, that is when max(N(F), N(R)) is not high enough (less than b). In practice, it is advisable to act cautiously, and to break the egg in a spare cup in case of serious doubt. It sounds like a more realistic advice than the one prescribed on the basis of the most likely state (like in [3]). Indeed, in the latter case, in spite of R (resp. F) being slightly certain, the chosen act is TA (resp. BIO), whose results may be not so good. In case of ignorance Boutilier's strategy gives no advice since the utility vectors (1, 0) for BIO, (d, b) for BAC and (a, c) for TA are incomparable. Results would be unchanged if "nothing to eat" had a non-zero utility strictly less than a.

3 THE VON NEUMANN-MORGENSTERN FRAMEWORK FOR UTILITY

Let us recall the axiomatic framework of Von Neumann and Morgenstern utility. Let X denote a set (of consequences). It is assumed that an objective probability distribution P on the state space S is known. Each act d carries over to X as a set of uncertain consequences called a lottery: $\{P_d(x) = P(d^{-1}(x)), x \in X\}$. $\mathbb{P}(X)$ denotes the set of lotteries $\{P_d, d \in X^S\}$ that is equivalent to the set of probability distributions on X. The set of lotteries is closed under convex mixing:

$$P, Q \in \mathbb{P}(X) \Rightarrow (P, \alpha, Q) = \alpha P + (1-\alpha)Q \in \mathbb{P}(X)$$

The problem is to numerically represent a prescribed ordering of lotteries given by an agent. One of the possible axiom systems for the Von Neumann and Morgenstern type utility is:

VNM1: $\mathbb{P}(X)$ is equipped with a complete preordering structure $\succcurlyeq$

VNM2: Independence (~ denotes indifference)
$P \sim Q \Rightarrow \forall \alpha \in [0,1], (P, \alpha, R) \sim (Q, \alpha, R)$

VNM3: Convexity
$P \succ Q \Rightarrow P \succ (P, \alpha, Q) \succ Q$, if $\alpha \in (0,1)$

VNM4: Continuity
$P \succ Q \succ R \Rightarrow \exists \alpha: Q \sim (P, \alpha, R)$

VNM5: Reduction of lotteries
$((P, \alpha, Q), \beta, Q) \sim (P, \alpha \cdot \beta, Q)$

The theorem below gives foundations to expected utility:

Theorem (Von Neumann - Morgenstern). There exists a function $u: \mathbb{P}(X) \to \mathbb{R}$ such that $P \succcurlyeq Q \Leftrightarrow u(P) \geq u(Q)$ and $u(P, \alpha, Q) = \alpha \cdot u(P) + (1-\alpha) \cdot u(Q)$. Moreover u is unique up to a linear transformation.

We now provide similar foundations for qualitative utility theory using possibility theory and qualitative scales.

Recall that V will denote a finite linear plausibility scale where inf(V) = 0, sup(V) = 1 and Pi(X) denotes the set of consistent (normal) possibility distributions on X (now assumed to be *finite*) over V, i.e., $Pi(X) = \{\pi: X \to V \mid \exists x \in X : \pi(x) = 1\}$. For the sake of simplicity, we shall use x for denoting both an element belonging to X and the possibility distribution π_x on X such that $\pi_x(x) = 1$ and $\pi_x(z) = 0$ for $z \neq x$. In general, we shall also denote by A both a subset $A \subseteq X$ and the possibility distribution π_A on X such that $\pi_A(x) = 1$ if $x \in A$ and $\pi_A(x) = 0$ otherwise. With this convention, we can consider X as included in Pi(X).

We already introduced *qualitative binary lotteries* $(\lambda/x, \mu/y)$. More generally using the notation $(\lambda_1/x_1, \ldots, \lambda_p/x_p)$, with $\lambda_i \in V$ and $\max_i(\lambda_i) = 1$, any consistent possibility distribution π on X can be seen as a multiple consequence qualitative lottery taking $\lambda_i = \pi(x_i)$. Note that the term lottery is not really appropriate here since $(\lambda_1/x_1, \ldots, \lambda_p/x_p)$ represents the qualitative description of an epistemic state, and the agent is now supposed to compare such epistemic states.

The so-called *Possibilistic mixture* (similar to a compound lottery, see [9]) is an operation defined on Pi(X) which combines two possibility distributions $\pi 1$ and $\pi 2$ into a new one, denoted $(\lambda/\pi 1, \mu/\pi 2)$, with

$\lambda, \mu \in V$ and $\max(\lambda, \mu) = 1$, and it is defined as:

$$(\lambda/\pi 1, \mu/\pi 2)(x) = \max(\min(\lambda, \pi 1(x)), \min(\mu, \pi 2(x))).$$

We shall build a finite linearly ordered scale of preference (or utility) U, with sup(U) = 1 and inf(U) = 0 for representing the ordering of epistemic states and a utility function u: $X \rightarrow U$ that assigns to each consequence of X a preference level of U and that extends to epistemic states.

The following axioms were proposed in [11] for a "rational" preference relation $\sqsubseteq$ on Pi(X) to be represented by a (pessimistic) qualitative utility (caution: $\pi \sim \pi'$ means $\pi' \sqsubseteq \pi$ and $\pi \sqsubseteq \pi'$):

- **Ax1**: $\sqsubseteq$ is a complete pre-order (i.e., $\sqsubseteq$ is reflexive, transitive and complete).
- **Ax2**: If A is a crisp subset of X then there is $x \in A$ such that $x \sim A$.
- **Ax3** (uncertainty aversion): $\pi \leq \pi' \Rightarrow \pi' \sqsubseteq \pi$.
- **Ax4** (independence): $\pi 1 \sim \pi 2 \Rightarrow (\lambda/\pi 1, \mu/\pi) \sim (\lambda/\pi 2, \mu/\pi)$.
- **Ax5** (reduction of lotteries): $(\lambda/x, \mu/(\alpha/x, \beta/y)) \sim (\max(\lambda, \min(\mu, \alpha))/x, \min(\mu, \beta)/y)$.
- **Ax6** (continuity): $\pi' \sqsubseteq \pi \Rightarrow \exists\, \lambda \in V$ such that $\pi' \sim (1/\pi, \lambda/X)$.

Axiom Ax1 makes it possible to represent utility on a totally ordered scale and is the same as VNM1 . Axioms Ax4, Ax5 and Ax6 are counterparts of axioms VNM2, VNM5, VNM4. Axiom Ax5 reduces higher order lotteries to standard ones in the style of possibilistic mixtures (see Figure 1 below).

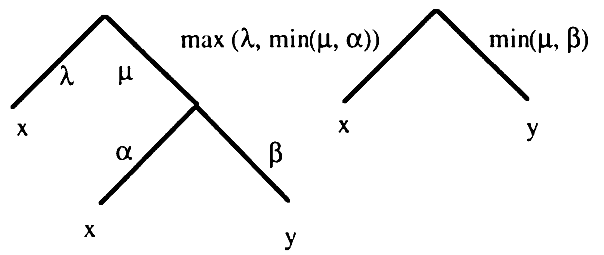


Figure 1: Possibilistic Mixture

The uncertainty aversion axiom states that the less informative is π, the more uncertain is the consequence: so, the worst epistemic state is total ignorance. So this axiom expresses an aversion for a lack of information. It plays the same role as the convexity axiom VNM3. Continuity says that if π' is less preferred than π, π' is preferentially equivalent to having some uncertainty about π.

Ax2 violated by expected utility, suggests that contrary to it, the pessimistic utility is *not* based on the idea of average and repeated decisions, but makes sense for one-shot decisions. Ax2 expresses that when the agent believes that the state lies in A and decision is put to work, then the state will be some x in A, and the benefit of the decision will indeed be the one in state x. It comes down to rejecting the notion of mean value. In fact lottery A is then equivalent to the worst consequence in A.

An interesting property of the preference relation $\sqsubseteq$ is, due to Ax1, Ax2 and Ax3, that the extremal elements of (X, $\sqsubseteq$) are maximal and minimal elements of (Pi(X), $\sqsubseteq$) as well.

Lemma 1. If $\sqsubseteq$ verifies axioms Ax1, Ax2 and Ax3, and $\underline{x}$ and $\overline{x}$ are a minimal, and a maximal element of X, respectively, then the following equivalences holds: $\underline{x} \sim (1/\overline{x}, 1/\underline{x}) \sim X$. Moreover $\underline{x}$ and $\overline{x}$ are the minimal and maximal elements of Pi(X) respectively.

Proof. Let us prove first the equivalences $\underline{x} \sim X \sim (1/\overline{x}, 1/\underline{x})$. Ax1 guarantees that $\underline{x}$ and $\overline{x}$ exist. By the uncertainty aversion axiom, it is clear that X is a minimal element of Pi(X), so it is $X \sqsubseteq \underline{x}$. But by Ax2 there exists $x_0 \in X$ such that $x_0 \sim X$, but since $\underline{x}$ is minimal, $\underline{x} \sqsubseteq x_0$, and thus it must be $\underline{x} \sim X$. Furthermore, on Pi(X) we have $\underline{x} \leq (1/\overline{x}, 1/\underline{x}) \leq X$ (specificity pointwise ordering) , and again by Ax3, $X \sqsubseteq (1/\overline{x}, 1/\underline{x}) \sqsubseteq \underline{x}$, and thus $\underline{x} \sim X \sim (1/\overline{x}, 1/\underline{x})$.
On the other hand, for any $\pi \in$ Pi(X), since π is normalised, there exists x such that $\pi(x) = 1$. So, we have $x \leq \pi$ and therefore $\pi \sqsubseteq x$, but since $\overline{x}$ is maximal in X, it is $x \sqsubseteq \overline{x}$, and thus $\pi \sqsubseteq \overline{x}$. □

Another interesting formulation of the continuity of the preference ordering is the following one:

- **Ax6***: For all $\pi \in$ Pi(X) there exists $\lambda \in V$ such that $\pi \sim (1/\overline{x}, \lambda/\underline{x})$

where $\overline{x}$ and $\underline{x}$ are a maximal and a minimal element of X w.r.t. $\sqsubseteq$ respectively, which will be very useful later. Indeed, it can be proved that, in the context of the rest of the axioms, axiom Ax6 is equivalent to Ax6*.

Proof. $\leftarrow$) Suppose Ax6* holds and let π, π' be such that $\pi' \sqsubseteq \pi$. We have two cases:
(i) $\pi' \sim \pi$. Then $\pi' \sim (1/\pi, 0/X)$.
(ii) $\pi' \sqsubset \pi$. By hypothesis there exists $\lambda, \lambda' \in V$ such that $\pi \sim (1/\overline{x}, \lambda/\underline{x})$ and $\pi' \sim (1/\overline{x}, \lambda'/\underline{x})$, since $\pi' \sqsubset \pi$ we have that $(1/\overline{x}, \lambda'/\underline{x}) \sqsubset (1/\overline{x}, \lambda/\underline{x})$, and by Ax3 it is $\lambda' > \lambda$. Now, taking into account that $X \sim \underline{x}$, the independence axiom and reducing lotteries we obtain that $(1/\pi, \lambda'/X) \sim (1/(1/\overline{x}, \lambda/\underline{x}), \lambda'/\underline{x}) = (1/\overline{x}, \max(\lambda', \lambda)/\underline{x})$. Since $\lambda' > \lambda$, $\max(\lambda', \lambda) = \lambda'$, and then $(1/\overline{x}, \max(\lambda', \lambda)/\underline{x}) = (1/\overline{x}, \lambda'/\underline{x}) \sim \pi'$. Therefore Ax6 also holds.

$\rightarrow$) Suppose now that Ax6 holds. For any π, we have that $\pi \sqsubseteq \overline{x}$. Then, by hypothesis, there exists λ such that $\pi \sim (1/\overline{x}, \lambda/X)$, and thus $\pi \sim (1/\overline{x}, \lambda/\underline{x})$. This proves that Ax6* also holds. □

4 REPRESENTATION OF QUALITATIVE UTILITIES

We propose next an improved set of axioms that characterise pessimistic qualitative utilities, and the corresponding axiomatics for the optimistic criterion, providing new proofs for their representation theorems. Actually, the set of axioms introduced in Section 3 is not minimal. The axiom on reduction of lotteries (Ax5) is unnecessary if we take the definition of possibilistic lotteries for granted. Namely, it is easy to show that the equality (in the sense of possibility distributions)

$(\lambda/x, \mu/(\alpha/x, \beta/y)) = (\max(\lambda, \min(\mu, \alpha))/x, \min(\mu, \beta)/y)$

always holds (the same remark applies to the VNM axiomatics (VNM4) if the notion of probabilistic mixture is acknowledged, see Herstein and Milnor [17]). On the other hand, Axiom Ax2 is also redundant since it follows from the rest of the axioms.

Lemma 2. Axioms Ax1, Ax4 and Ax6 imply axiom Ax2.

Proof. Suppose that $A = \{x_1, x_2\}$ with $x_1 \sqsubseteq x_2$, by Ax6 we have that there exists λ s.t. $x_1 \sim (1/x_2, \lambda/X)$, and applying Ax1, Ax4 we obtain $A = (1/x_1, 1/x_2) \sim (1/(1/x_2, \lambda/X), 1/x_2) = (1/x_2, \lambda/X) \sim x_1$. The case when A has n elements is an easy generalisation. □

The above discussion has led us to propose this new set of axioms:

- **A1***: $\sqsubseteq$ is a total pre-order .
- **A2*** (uncertainty aversion): if $\pi \leq \pi' \Rightarrow \pi' \sqsubseteq \pi$.
- **A3*** (independence): $\pi_1 \sim \pi_2 \Rightarrow (\lambda/\pi_1, \mu/\pi) \sim (\lambda/\pi_2, \mu/\pi)$.
- **A4***: For all $\pi \in \mathrm{Pi}(X)$ there exists $\lambda \in V$ such that $\pi \sim (1/\overline{x}, \lambda/\underline{x})$, where $\overline{x}$ and $\underline{x}$ are a maximal and a minimal element of X w.r.t. $\sqsubseteq$ respectively.

Next we show that the preference ordering on Pi(X) induced by our qualitative pessimistic utility satisfies the above set of axioms. Let $u: X \to U$ be a preference function such that $u^{-1}(1) \neq \emptyset \neq u^{-1}(0)$, and let $h: V \to U$ be an order preserving function relating both scales V and U such that $h(0) = 0$, $h(1) = 1$ and assume that h is onto. In practice, if h is not onto we can make it onto by adding elements to V, thus refining the uncertainty scale, and mapping them to elements outside h(V) in such a way that the ordering is preserved. For any $\pi \in \mathrm{Pi}(X)$, consider again the qualitative utility $QU^-(\pi) = \min_{x \in X} \max(n_U(\pi^*(x)), u(x))$ where $\pi^*(x) = h(\pi(x))$. Notice that QU^-, restricted to X, coincides with the utility function u, i.e., $QU^-(x) = u(x)$, for all $x \in X$.

$$\begin{tikzcd}
X \arrow[r, "\pi"] \arrow[d, "u"'] & V \arrow[d, "h"] \arrow[dl, "n"'] \\
U \arrow[r, leftrightarrow, "n_U"'] & U
\end{tikzcd}$$

Figure 2: Diagram of the different mappings

Let us introduce the order-reversing mapping $n: V \to U$ defined as $n(\lambda) = n_U(h(\lambda))$. It verifies $n(0) = 1$, $n(1) = 0$. Actually, since $n_U{}^2$ is the identity in U, the mapping h can also be defined from n, namely $h(\lambda) = n_U(n(\lambda))$. See Fig. 2. Using n instead of h, the qualitative utility may be equivalently expressed as:

$$QU^-(\pi) = \min_{x \in X} \max(n(\pi(x)), u(x)). \tag{3}$$

It is interesting to notice that QU^- preserves the possibilistic mixture in the sense that $QU^-((\lambda/\pi_1, \mu/\pi_2)) = \min\{\max(n(\lambda), QU^-(\pi_1)), \max(n(\mu), QU^-(\pi_2))\}$. This is again the median of three terms: $QU^-(\pi_1)$, $QU^-(\pi_2)$ and $\min(n(\lambda), n(\mu))$, where $\max(\lambda,\mu) = 1$. It behaves like the classical EUT, changing median by weighted mean, as shown above for the Von Neumann - Morgenstern model.

Lemma 3. Let $\preccurlyeq_u$ be the preference ordering on Pi(X) induced by QU^-, i.e. $\pi \preccurlyeq_u \pi'$ iff $QU^-(\pi) \leq QU^-(\pi')$. Then $\preccurlyeq_u$ verifies axioms A1*, A2*, A3* and A4*.

Proof. Axioms A1* and A2* are easily verified and A3* is a consequence of the fact that QU^- preserves the possibilistic mixture. Thus we only check axiom A4*. We have to prove that $\forall \pi \in \mathrm{Pi}(X)$, there exists λ such that $QU^-(\pi) = QU^-((1/\overline{x}, \lambda/\underline{x}))$, where $\overline{x}$, $\underline{x}$ are a maximal and a minimal element of X w.r.t. $\preccurlyeq_u$. Since we are assuming $u^{-1}(1) \neq \emptyset \neq u^{-1}(0)$, it must be the case that $u(\underline{x}) = 0$ and $u(\overline{x}) = 1$ and thus, by the possibilistic mixture preservation by QU^- we have that $QU^-((1/\overline{x}, \lambda/\underline{x})) = \min\{\max(n(1), QU^-(\overline{x})), \max(n(\lambda), QU^-(\underline{x}))\} = n(\lambda)$. Since h is onto, n is onto as well and it is $u(X) \subseteq U = n(V)$; therefore, for any $\lambda \in n^{-1}(QU^-(\pi))$ we have that $QU^-(\pi) = n(\lambda) = QU^-((1/\overline{x}, \lambda/\underline{x}))$. □

Now we can show that the preference orderings on epistemic states satisfying the axioms proposed can always be represented by a pessimistic qualitative utility of the type of QU^-.

Representation Theorem 1. *(Pessimistic Utility)* A preference relation $\sqsubseteq$ on Pi(X) satisfies axioms A1*, A2*, A3* and A4* if, and only if, there exist

(i) a linearly ordered utility scale U with $\inf(U) = 0$ and $\sup(U) = 1$,
(ii) a preference function $u: X \to U$ such that $u^{-1}(1) \neq \emptyset \neq u^{-1}(0)$, and
(iii) an onto order reversing function $n: V \to U$ such that $n(0) = 1$ and $n(1) = 0$,

in such a way that $\pi' \sqsubseteq \pi$ iff $\pi' \preccurlyeq_u \pi$, where $\preccurlyeq_u$ is the ordering in Pi(X) induced by the qualitative utility QU^- defined as $QU^-(\pi) = \min_{x \in X} \max(n(\pi(x)), u(x))$.

Proof. The "if" part corresponds to the preceeding Lemma. As for the "only if" part, we structure the proof in the following three steps. In step (I) we define the utility scale U and an order reversing (and surjective) function n from V to U. In step (II) we define a function $QU^-: \mathrm{Pi}(X) \to U$ representing $\sqsubseteq$, i.e. such that $QU^-(\pi) \leq QU^-(\pi')$ iff $\pi \sqsubseteq \pi'$. Finally in step (III) we prove that $QU^-(\pi) = \min_{x \in X} \max(n(\pi(x)), u(x))$, where $u: X \to U$ is the restriction of QU^- on X. Now we develop these steps.

I) First of all, notice that $\sqsubseteq$ stratifies Pi(X) in a linearly ordered set of classes of equivalently preferred distributions ($\pi' \in [\pi]$ iff $\pi \sim \pi'$). The number of classes is just the number of levels needed to rank order the set of distributions. Therefore we take as utility scale U the quotient set Pi(X)/~ together with the natural (linear) order $[\pi] \le [\pi']$ iff $\pi \sqsubseteq \pi'$. Denote by 1 and 0 the maximum and minimum elements of Pi(X)/~, i.e., of U. By Lemma 1, if $\overline{x}$ and $\underline{x}$ are a maximal and minimal elements of (X, $\sqsubseteq$) respectively, then clearly $[\overline{x}] = 1$ and $[\underline{x}] = 0$. Denote by π_λ the possibility distribution corresponding to the qualitative lottery $(1/\overline{x}, \lambda/\underline{x})$ and define the order reversing function n: $V \to U$ as $n(\lambda) = [\pi_\lambda]$.

Observe that, since $(1/\overline{x}, 1/\underline{x}) \sqsubseteq \underline{x}$, $n(1) = [(1/\overline{x}, 1/\underline{x})] = [\underline{x}] = 0$ and $n(0) = [(1/\overline{x}, 0/\underline{x})] = [\overline{x}] = 1$. We verify now that n actually reverses the order. Let $\lambda < \lambda'$, then $\pi_\lambda \le \pi_{\lambda'}$, so using A2* we have $\pi_{\lambda'} \sqsubseteq \pi_\lambda$, then by definition $[\pi_{\lambda'}] \le [\pi_\lambda]$ i.e., $n(\lambda') \le n(\lambda)$. Observe that, by construction, n is onto, indeed, for any $\pi \in$ Pi(X), A4* guarantees that there exists λ s.t. $\pi_\lambda \sim \pi$, so $n(\lambda) = [\pi]$.

II) So far we have determined U and **n**. Now we define the qualitative function QU^- on Pi(X) in two steps.

(i) First define $QU^-(1/\overline{x}, \lambda/\underline{x}) = n(\lambda)$. It is easy to check that $\pi_\lambda \sqsubseteq \pi_{\lambda'}$ iff $QU^-(\pi_\lambda) \le QU^-(\pi_{\lambda'})$. Indeed, $\pi_\lambda \sqsubseteq \pi_{\lambda'}$ iff $[\pi_\lambda] \le [\pi_{\lambda'}]$ iff $n(\lambda) \le n(\lambda')$ iff $QU^-(\pi_\lambda) \le QU^-(\pi_{\lambda'})$. So, restricted to lotteries of type π_λ, QU^- represents $\sqsubseteq$.

(ii) We extend QU^- to any lottery as follows. Since for any π, A4* guarantees that $\exists\, \lambda$ such that $\pi \sim (1/\overline{x}, \lambda/\underline{x})$, we define $QU^-(\pi) = n(\lambda)$. Notice that QU^- is well defined: suppose there exists $\mu \ne \lambda$ such that $\pi \sim (1/\overline{x}, \mu/\underline{x})$. But since $(1/\overline{x}, \mu/\underline{x}) \sim (1/\overline{x}, \lambda/\underline{x})$ then $[\pi_\lambda] = [\pi_\mu]$, so $n(\lambda) = n(\mu)$. Finally it is easy to check that QU^- represents $\sqsubseteq$. This is due to the fact that any π is equivalent to some π_λ, and by (i) QU^- represents $\sqsubseteq$ over the π_λ's.

III) Now we define u: $X \to U$ as $u(x) = QU^-(\pi_x)$. Notice that $u(\overline{x}) = 1$ and $u(\underline{x}) = 0$, and thus, $u^{-1}(1) \ne \emptyset \ne u^{-1}(0)$. It remains to prove that $QU^-(\pi) = \min_{x \in X} \max(n(\pi(x)), u(x))$. To verify this we will prove the following equalities:

• $QU^-(1/x, \lambda/y) = \min(u(x), \max(n(\lambda), u(y)))$. Indeed, A4* guarantees that $\exists\, \mu$ such that $x \sim (1/\overline{x}, \mu/\underline{x})$ and $\exists\, \gamma$ such that $y \sim (1/\overline{x}, \gamma/\underline{x})$ —remember that $QU^-(x) = u(x) = n(\mu)$ and $QU^-(y) = u(y) = n(\gamma)$—, so using A3*, we have $(1/x, \lambda/y) \sim (1/(1/\overline{x}, \mu/\underline{x}), \lambda/(1/\overline{x}, \gamma/\underline{x}))$, and reducing lotteries we obtain $(1/x, \lambda/y) \sim (\max(1, \lambda)/\overline{x}, \max(\mu, \min(\lambda, \gamma))/\underline{x})) = (1/\overline{x}, \max(\mu, \min(\lambda, \gamma))/\underline{x})$. Therefore, $QU^-(1/x, \lambda/y) = n(\max(\mu, \min(\lambda, \gamma))) = \min(n(\mu), \max(n(\lambda), n(\gamma))) = \min(u(x), \max(n(\lambda), u(y)))$.

• $QU^-(\max(\pi_1, \pi_2)) = \min(QU^-(\pi_1), QU^-(\pi_2))$. By A4*, $\exists\, \mu$ such that $\pi_1 \sim (1/\overline{x}, \mu/\underline{x})$ and $\exists\, \gamma$ such that $\pi_2 \sim (1/\overline{x}, \gamma/\underline{x})$. Then, using A3*, we have: $\max(\pi_1, \pi_2) = (1/\pi_1, 1/\pi_2) \sim (1/(1/\overline{x}, \mu/\underline{x}), 1/(1/\overline{x}, \gamma/\underline{x}))$, i.e., $\max(\pi_1, \pi_2) \sim (1/\overline{x}, \max(\mu, \gamma)/\underline{x}))$. Therefore, $QU^-(\max(\pi_1, \pi_2)) = n(\max(\mu, \gamma)) = \min(n(\mu), n(\gamma)) = \min(QU^-(\pi_1), QU^-(\pi_2))$. More generally, we will have $QU^-(\max_{i=1,p} \pi_i) = \min_{i=1,p} QU^-(\pi_i)$.

• $QU^-(\pi) = \min_{i=1,p} \max(n(\pi(x_i)), u(x_i))$. As π is normalised there exists $x_j \in X$ such that $\pi(x_j) = 1$. Without loss of generality assume $j = 1$. Then let $\pi_i = (1/x_1, \pi(x_i)/x_i)$. Since $\pi = \max_{i=1,p} \pi_i$, we have:

$$\begin{aligned} QU^-(\pi) &= QU^-(\max_{i=1,p} \pi_i) = \min_{i=1,p} QU^-(\pi_i) = \\ &= \min_{i=1,p}\{\min(u(x_1), \max(n(\pi(x_i)), u(x_i))\} =^1 \\ &= \min_{i=1,p} \max(n(\pi(x_i)), u(x_i)). \end{aligned}$$

This ends the proof of the theorem. □

Now if we move to an optimistic preference criterium, we have to change the uncertainty aversion axiom A2* by a *uncertainty-prone* postulate

• **A2⁺**: if $\pi \le \pi'$ then $\pi \sqsubseteq \pi'$

and to adequately modify the continuity axiom A4* into

• **A4⁺**: for all $\pi \in$ Pi(X), there exists $\lambda \in V$ such that $\pi \sim (\lambda/\overline{x}, 1/\underline{x})$, where $\overline{x}$ and $\underline{x}$ are a maximal and a minimal element of (X, $\sqsubseteq$).

In an analogous way to the pessimistic case, one could equivalently use the axiom

• **Ax6'** (continuity):
$\pi' \sqsubseteq \pi \Rightarrow \exists\, \lambda \in V$ such that $\pi \sim (1/\pi', \lambda/X)$

instead of A4⁺. We omit here further details and just state that the set of axioms A1*, A2⁺, A3* and A4⁺ faithfully characterise the preference orderings induced by an optimistic qualitative utility, which is dual to the pessimistic one.

Representation Theorem 2. *(Optimistic Utility)* A preference relation (Pi(X), $\sqsubseteq$) satisfies axioms A1*, A2⁺, A3* and A4⁺, if and only if there exist

(i) a linearly ordered utility scale U, with inf(U) = 0, sup(U) = 1,
(ii) a preference function u: $X \to U$ such that $u^{-1}(1) \ne \emptyset \ne u^{-1}(0)$, and
(iii) an onto order preserving function h: $V \to U$ such that $h(0) = 0$, $h(1) = 1$,

in such a way that it holds: $\pi' \sqsubseteq \pi$ iff $\pi' \lhd_u \pi$, where $\lhd_u$ is the ordering on Pi(X) induced by the qualitative utility $QU^+(\pi) = \max_{x \in X} \min(h(\pi(x)), u(x))$.

Proof. The proof is very analogous to the one for pessimistic utility and we only sketch the proof for the "only if" part. For the same reasons as before we choose U = Pi(X)/~. Again, if $\underline{x}$ and $\overline{x}$ denote a minimal and a maximal element of (X, $\sqsubseteq$) respectively, $[\overline{x}]$ and $[\underline{x}]$ will be the 1 and 0 of U. We define h: $V \to U$ as $h(\lambda) = [(\lambda/\overline{x}, 1/\underline{x})]$. Observe that $h(1) = [1/\overline{x}, 1/\underline{x})] = [\overline{x}] = 1$, and $h(0) = [(0/\overline{x}, 1/\underline{x})] = [\underline{x}] = 0$. Moreover, due to the uncertainty-prone axiom it is easy to check that h is order

1 Note that $\pi(x_1)=1$, so $u(x_1) = \max(u(x_1), n(\pi(x_1)))$.

preserving. From that we only sketch the main steps of the proof:

- Define $QU^+(\lambda/\overline{x}\ ,\ 1/\underline{x}) = h(\lambda)$.
- Verify that if $\pi_\lambda \sqsubseteq \pi_{\lambda'}$ then $QU^+(\pi_\lambda) \leq QU^+(\pi_{\lambda'})$.
- Extend QU^+ for any π, due to axiom A4$^+$.
- Define $u(x) = QU^+(\pi_x)$.
- Verify that $QU^+(1/x, \lambda/y) =$
  $= \max(u(x), \min(h(\lambda), u(y)))$.
- Verify that $QU^+(\max(\pi_1, \pi_2)) =$
  $= \max(QU^+(\pi_1), QU^+(\pi_2))$.
- Verify that $QU^+(\pi) = \max_{x\in X} \min(h(\pi(x)), u(x))$.
- Verify that $\lhd u$ agrees with $\sqsubseteq$. □

In practice QU^+ is a very optimistic index which can be used for refining the ordering given by QU^-. Finally, we would like to stress that the qualitative utility functions QU^- and QU^+ are indeed "utility" functions in Pi(X) in the sense that they preserve the preference ordering and the "natural operation" of possibilistic mixture (PM) used to combine possibilistic lotteries or distributions. Indeed, let $\Phi_{\max} = \{(\alpha, \beta) \in V\times V \mid \max(\alpha, \beta) = 1\}$. If we consider the possibilistic mixture operation PM as the mapping

$$PM: Pi(X) \times Pi(X) \times \Phi_{\max} \rightarrow Pi(X)$$

defined by $PM(\pi, \pi'; \alpha, \beta) = (\alpha/\pi, \beta/\pi')$, then $QU^-(PM(\pi, \pi', \alpha, \beta)) = UM^-(QU^-(\pi), QU^-(\pi'); \alpha, \beta)$, where UM^- is the corresponding mixture in the preference scale U, $UM^-: U \times U \times \Phi_{\max} \rightarrow U$, defined by

$$UM^-(\mu, \mu'; \gamma, \delta) = \min(\max(n(\gamma), \mu), \max(n(\delta), \mu')). \quad (4)$$

That is to say, QU^- is a morphism between the structure of possibilistic lotteries and the structure of the qualitative preference scale. For the optimistic qualitative utility we have analogous results: QU^+ preserves the order and the mixture operation with respect to the operation UM^+: $U \times U \times \Phi_{\max} \rightarrow U$, defined as

$$UM^+(\mu, \mu'; \gamma, \delta) = \max(\min(h(\gamma), \mu), \min(h(\delta), \mu')), \quad (5)$$

in the sense that it holds

$$QU^+(PM(\pi, \pi', \alpha, \beta)) = UM^+(QU^+(\pi), QU^+(\pi'); \alpha, \beta).$$

5 GENERALIZATION TO MAX-T POSSIBILISTIC MIXTURES

The possibilistic mixture operation considered so far to combine possibilistic lotteries is a max-min combination: $(\alpha/\pi_1, \beta/\pi_2) = \max(\min(\alpha, \pi_1), \min(\beta, \pi_2))$. Possibilistic mixtures, definable as consensus functions on $\perp$-*decomposable measures*, $\perp$ being a t-conorm operation[2], have been studied in [9]. It is shown there that for possibility measures, i.e., max-decomposable measures, an admissible class of mixture operations is obtained by defining

$$M_T(\pi, \pi'; \alpha, \beta) = \max(\alpha T \pi, \beta T \pi') \quad (6)$$

where T is a *t-norm* operation[3] and $\max(\alpha,\beta) = 1$. A particular case is thus to take T = min. In order to encompass this extended kind of possibilistic mixture operations in the qualitative decision model we have considered the modified axiom set $AX_T = \{A1^*, A2^*, A3^*_T, A4^*_T\}$, where

- **A3$^*_T$** (independence):
  $\pi_1 \sim \pi_2 \Rightarrow M_T(\pi_1, \pi; \alpha, \beta) \sim M_T(\pi_2, \pi; \alpha, \beta)$
- **A4$^*_T$**: For all $\pi \in Pi(X)$ there exists $\lambda \in V$ such that $\pi \sim M_T(\overline{x}, \underline{x}; 1, \lambda)$, where $\overline{x}$ and $\underline{x}$ are a maximal and a minimal element of X with respect to $\sqsubseteq$ respectively.

Now, the reduction of lotteries follows the next rule:

$$M_T(M_T(\pi_1, \pi_2; \lambda_1, \lambda_2), M_T(\pi_1, \pi_2; \mu_1, \mu_2), \alpha, \beta) =$$
$$= M_T(\pi_1, \pi_2; \max(\alpha T \lambda_1, \beta T \mu_1), \max(\alpha T \lambda_2, \beta T \mu_2)).$$

It is not difficult to verify that we may obtain results for this axiomatic system that are very analogous to the results obtained in the previous section. The resulting extension of the qualitative utility QU^- turns out to be the following expresion:

$$GQU^-(\pi) = \min_{x_i\in X} n(\pi(x_i)\ T\ \lambda_i), \quad (7)$$

where $n(\lambda_i) = u(x_i)$, the order reversing mapping $n: V \rightarrow U$ being as usual, but further verifying a *coherence* condition w.r.t. T: $n(\lambda) = n(\mu) \Rightarrow n(\alpha T \lambda) = n(\alpha T \mu)$, for all $\alpha, \lambda, \mu \in V$. This condition guarantees the correctness of the above definition. Notice that either when T = min (then (7) retrieves (3)) or when n is injective this condition is satisfied. For the preference orderings induced by these generalized qualitative utilities we have a representation theorem like in the previous section.

Representation Theorem. A preference relation $\sqsubseteq$ on Pi(X), equipped with the mixture operation M_T, satisfies the axiom set AX_T if and only if there exist

(i) a linearly ordered preference scale U with inf(U) = 0 and sup(U) = 1,
(ii) a preference function $u: X \rightarrow U$ such that $u^{-1}(1) \neq \emptyset \neq u^{-1}(0)$,
(iii) an onto order reversing function $n: V \rightarrow U$ such that $n(0) = 1$, $n(1) = 0$, satisfying $n(\lambda) = n(\mu) \Rightarrow n(\alpha T \lambda) = n(\alpha T \mu)$, $\forall\ \alpha, \lambda, \mu \in V$,

in such a way that it holds: $\pi' \sqsubseteq \pi$ iff $\pi' \preccurlyeq_u \pi$,

where $\preccurlyeq_u$ is the ordering in Pi(X) induced by the qualitative utility GQU^- defined as follows: $GQU^-(\pi) = \min_{x_i\in X} n(\pi(x_i)\ T\ \lambda_i)$, where $n(\lambda_i) = u(x_i)$.

An analogous result for an optimistic behaviour may be obtained as well. The corresponding utility function is then, denoting $h(\lambda_i) = u(x_i)$:

[2]A measure $g: 2^X \rightarrow V$ is $\perp$-decomposable if $g(A\cup B) = g(A) \perp g(B)$ when $A\cap B = \emptyset$, where $\perp$ is a non-decreasing, associative and commutative binary operation on V, with $v \perp 0 = v$ and $v \perp 1 = 1$ for all $v \in V$.

[3]A t-norm operation T on V is a non-decreasing, associative and commutative binary operation on V, verifying $vT0 = 0$ and $vT1 = v$ for all $v \in V$.

$$GQU^{+}(\pi) = \max_{x_i \in X} h(\pi(x_i) \top \lambda_i)$$

N.B. Suppose that $V = \{2^{-n}, n \in \mathbb{N}\} = U$, and h is identity. Then choosing the t-norm $\top$ as product, it is possible to connect the utility function $GQU^{+}(\pi)$ to the integer-valued utility of Pearl [21] and to classical utility theory via infinitesimals. Namely, if $u(x) = \varepsilon^{-i}$, $u(y) = \varepsilon^{-j}$, i and j being relative integers, and ε being an infinitesimal quantity, consider the probabilistic lottery $P = (y, \varepsilon^{n}, x)$. Then the classical utility U(P) is:

$$U(P) = (1 - \varepsilon^{n})\cdot u(x) + \varepsilon^{n}\cdot u(y) = \varepsilon^{\min(-i,\, n-j)}.$$

Consider the order-equivalent quantity $2^{-\min(-i,\, n-j)}$. Let $u'(x) = 2^{i}$, $u'(y) = 2^{j}$, $\pi(x) = 1$, $\pi(y) = 2^{-n}$. Then the infinitesimal utility U(P), or equivalently the Pearl-like integer-valued utility $\min(-i, n-j)$, can be represented by $GQU^{+}(\pi) = \max(\pi(x) \top u'(x), \pi(y) \top u'(y))$.

6 CASE-BASED DECISION

The decision model under uncertainty described so far relies on a possibilistic representation of the uncertainty concerning belief states. Such a representation can be made explicit for instance if (uncertain) generic knowledge and information is available under the form of a possibilistic knowledge base [12].

Suppose now we have only partial information about the possible consequences of decisions by having stored the performance of decisions taken in different past situations as a set M of triples (*situation, decision, consequence*).

Let us mention that, in such a framework, Gilboa and Schmeidler [16] have recently proposed a case-based decision model where the decision-maker, in face of a new situation s_0, is supposed to choose a decision *d* which maximizes a counterpart of classical expected utility, namely

$$U_{s_0,M}(d) = \sum_{(s,d,x)\in M} \mathcal{S}(s_0,s)\cdot u(x) \qquad (8)$$

where $\mathcal{S}$ is a non-negative function which estimates the similarity of situations and u provides a numerical utility for each consequence x. However, we cannot assume $\sum_{(s,d,x)\in M} \mathcal{S}(s_0,s) = 1$, for any s_0, which differs from the situation in classical utility theory. Gilboa and Schmeidler provide an axiomatic derivation of this U-maximization, within a formal model.

It is possible, and probably more realistic to present the omelette story of Section 2 as a case-based decision problem. The memory consists of descriptions of eggs broken in the past by the agent, the decisions made about those eggs (BAC, BIO, TA), and the outcomes (one of the six ones in Table 1). Descriptions are in terms of attributes describing for instance the colour, the smell, the weight of the egg, etc. The decision made about a new egg for a new omelette can be based on the resemblance between the present egg and the past ones. If the the egg *looks* fresh (= is similar to the descriptions of past fresh eggs) then BIO, if the the egg *looks* rotten then TA, if the egg is only mildly fresh but not clearly rotten, or it is a new type of egg not encountered in the past then for instance BAC.

The approach we propose here is to estimate to what extent a consequence x can be considered plausible, in a current situation s_0 after taking a decision d, in terms of what extent the current situation s_0 is similar to situations in which x was experienced after taking the decision d. This amounts to assume, for each case (s, d, x) in a memory M, a principle stating that "*The more similar is s_0 to s, the more possible x is a consequence for s_0*". This kind of guiding meta-rule has been recently considered in [6] for case-based reasoning. According to that principle, given a memory of cases[4] M, if a similarity relation is available in the set of situations $\mathcal{S}: S \times S \to V$[5], the following possibility distribution $\pi_{d,s_0}: X \to V$ on the set of consequences can be assumed

$$\pi_{d,s_0}(x) = \max\{\mathcal{S}(s_0, s) \mid (s, d, x) \in M\}, \qquad (9)$$

where, by convention, we take $\max \emptyset = 0$. Then, given a utility function on the set of consequences $u: X \to U$, the utility $\mathcal{U}^{-}_{s_0}(d)$ of decision d can be estimated, in a conservative way, as $\mathcal{U}^{-}_{s_0}(d) = QU^{-}(\pi_{d,s_0})$, i.e.

$$\begin{aligned}\mathcal{U}^{-}_{s_0}(d) &= \min_{x\in X} \max(n(\pi_{d,s_0}(x)), u(x)) \qquad (10)\\ &= \min_{(s,d,x)\in M} \max(n(\mathcal{S}(s_0, s)), u(x)),\end{aligned}$$

where again $n: V \to U$ is an order reversing mapping such that $n(0) = 1$ and $n(1) = 0$. For the sake of simplicity we asume that the scales V and U are the same and the mapping n is just the involution on U. Maximizing this utility corresponds to look for decisions which for similar situations always gave good results. As already mentioned, $QU^{-}(\pi_{d,s_0})$ expresses a degree of inclusion, in this case, the degree of inclusion of the fuzzy set of situations which are similar to s_0 and where decision d was experienced, into the fuzzy set of situations where decision d led to good results. Indeed:

- $\mathcal{U}^{-}_{s_0}(d) = 1$ if $\{s \mid (s, d, x) \in M, \mathcal{S}(s,s_0) > 0\} \subseteq \{s \mid (s, d, x) \in M, u(x) = 1\}$
- $\mathcal{U}^{-}_{s_0}(d) = 0$ as soon as $\exists s$ such that $\mathcal{S}(s,s_0) = 1$, $(s, d, x) \in M$ and $u(x) = 0$.

Actually, $\mathcal{U}^{-}_{s_0}(d)$ is a rather drastic criterium since it requires that in *all* the situations similar to s_0, d led in good results. For instance, all past eggs similar to the present one were broken in the omelette and turned out to be fresh. A more "optimistic" behaviour consists in selecting decisions which led to a good result in at least one situation similar to s_0, by using the dual criteria $\mathcal{U}^{+}_{s_0}(d) = QU^{+}(\pi_{d,s_0})$, i.e.

$$\begin{aligned}\mathcal{U}^{+}_{s_0}(d) &= \max_{x\in X} \min(h(\pi_{d,s_0}(x)), u(x)) \qquad (11)\\ &= \max_{(s,d,x)\in M} \min(h(\mathcal{S}(s_0, s)), u(x)).\end{aligned}$$

See [6] for a different derivation of similar expressions of

[4] We may assume that cases are deterministic, i.e., if (s, d, x), (s, d, x') $\in$ M, then x=x', as in [16], but it is not compulsory.
[5] Here it is assumed that $\mathcal{S}$ is reflexive, i.e., $\mathcal{S}(s, s) = 1$, and symmetric, i.e. $\mathcal{S}(s, s') = \mathcal{S}(s', s)$.

$\mathcal{U}^-_{s_0}(d)$ and $\mathcal{U}^+_{s_0}(d)$. Thus $\mathcal{U}^+_{s_0}(d)$ is maximal as soon as there exists a case corresponding to a situation completely similar to s_0 where d led to an excellent result. For instance, you break the egg in the omelette if you remember at least one fresh egg similar to the one you hold. The pessimistic and optimistic decision rules differ from the Gilboa-Schmeidler rule in that they do not assume that results obtained with past eggs accumulate and, particularly, compensate. Using Gilboa-Schmeidler rule, a few bad experiences with a certain kind of egg very similar to the current one can be fully counterbalanced by sufficiently many half-fresh eggs of similar appearance. The pessimistic criterion suggests mistrusting these eggs and the optimistic one only partially tolerates them.

We have, however, to be very cautious if we want to apply this qualitative decision model: nothing prevents the distributions π_{d,s_0} from being non-normalized. And this may have undesirable consequences, such as the fact that the pessimistic utility $\mathcal{U}^-_{s_0}(d)$ may be higher than the optimistic utility $\mathcal{U}^+_{s_0}(d)$. Indeed when

$$height(\pi_{d,s_0}) = \max_{(s,d,x)\in M} \mathcal{S}(s,s_0) < 1, \quad (12)$$

it means that decision d has been never experienced on a situation completely similar to s_0. In particular when $\{s \mid (s, d, x) \in M, S(s,s_0) > 0\} = \emptyset$, we have $\mathcal{U}^-_{s_0}(d) = 1$ which is not satisfactory. In order to remedy this problem, we propose to modify our previous definition and let $\mathcal{U}^-_{s_0}(d) = \underline{QU}^-(\pi_{d,s_0})$, where

$$\underline{QU}^-(\pi_{d,s_0}) = \min(height(\pi_{d,s_0}), QU^-(\pi^*_{d,s_0})) \quad (13)$$

where π^*_{d,s_0} is a renormalized[6] version of π_{d,s_0}. Notice that when $height(\pi_{d,s_0}) = 1$, the original expression is retrieved. The rationale behind (13) is that our willingness to apply decision d in s_0 is upper bounded by the existence of situations completely similar to s_0 where decision d was experienced. Moreover π_{d,s_0} is renormalized in order to obtain a meaningful degree of inclusion. Thus, (13) corresponds to the expression of the compound condition "there exist situations similar to s_0 where decision d was used and the situations *which are the most similar to s_0* are among the situations where decision d led to good results". Note that the similarity is no longer estimated in an absolute manner, but in a relative way, hence the normalization.

Clearly, it would be also natural that the optimistic evaluation be all the greater as the decision d was never applied to situations similar to s_0 in the past (indeed in this case, the optimistic decision-maker is prone to experiencing new decisions on new situations he never met). This leads to define $\mathcal{U}^+_{s_0}(d) = \underline{QU}^+(\pi_{d,s_0})$, where

$$\underline{QU}^+(\pi_{d,s_0}) = \max(n(height(\pi_{d,s_0})), QU^+(\pi^*_{d,s_0})).(14)$$

Again, when $height(\pi_{d,s_0}) = 1$ we recover the original definition. Using the modified definitions, the inequality $\mathcal{U}^-_{s_0}(d) \le \mathcal{U}^+_{s_0}(d)$ is then preserved.

Notice that by ranking decisions by means of $\underline{QU}^-(\pi) = \min(height(\pi), QU^-(\pi^*))$ we extend the reference decision set in Section 4 from the set of normalized possibility distributions Pi(X) into the set $Pi^{ex}(X)$ of non necessarily normalized distributions on the set of consequences X. By doing that, the axiom set presented in Section 3 for the pessimistic utility is no longer sound. For instance the uncertainty aversion axiom A2* does not hold for the preference ordering induced by $\underline{QU}^-$. Actually, this ordering results from the combination of the orderings induced by the first component of $\underline{QU}^-(\pi)$ (i.e., $height(\pi)$) and by the second component (i.e., $QU^-(\pi^*)$), which act in opposite directions. The axiomatization of the preference ordering induced by utility functions of type $\underline{QU}^-$ can be achieved by adding a supplementary axiom specifically coping with subnormalized possibility distributions, namely

- **A5***: $\forall\pi, \exists \lambda$, such that $\pi \sim (1/\pi^*, \lambda/\underline{x})$,

where π^* is the renormalized version of π defined in footnote 6. The value λ is the transform of the $height(\pi)$ via the order-reversing map on V. What it says is that subnormal possibility distributions are viewed as normal ones tainted with a level of uncertainty all the greater as height(π) is small.

7 CONCLUSION

This paper offers an improved axiomatic justification of qualitative graded extensions of the celebrated maximin and maximax criteria for decision making under incomplete information, based on possibility theory, in the Von Neumann Morgenstern style. The only other very recent work that tries to generalize these criteria on the basis of Von Neumann - Morgenstern setting is by Lehmann [19], using infinitesimals.

The next step is to study the foundations of qualitative decision in a dynamic setting, taking conditioning into account. Namely how is qualitative utility revised upon learning that some event occurred that restricts the set of possible consequences. This problem has some connection with the one of subnormal possibility distributions. However it seems to be more naturally addressed in a Savage-like setting, where preference is directly defined on acts, than in the one of Von Neumann - Morgenstern, where the uncertainty theory is taken for granted. Lehmann [18] gives some interesting insight into conditional preference when the sure thing principle is not assumed from the start. Some work is going on towards the counterpart of Savage approach (in a static setting so far) for the qualitative decision theory proposed in this paper, partly based on a modification of the sure-thing principle (see [13] for preliminary results), and that might be pursued along this line.

Acknowledgements

The research has been partially supported by a CNRS-

[6] The renormalized distribution is defined as $\pi^*_{d,s_0}(x) = \pi_{d,s_0}(x)$ for $x \neq x_0$ and $\pi^*_{d,s_0}(x_0) = 1$, where x_0 is such that $\pi_{d,s_0}(x_0) = \max\{\pi_{d,s_0}(x) \mid x \in X\}$.

CSIC bilateral cooperation project. Besides, Adriana Zapico acknowledges partial support by a doctoral scolarship MUTIS and by the Universidad Nacional de Río Cuarto (Argentina). Lluís Godo has also been partially supported by the CICYT project SMASH (TIC96-1138-C04-01).

References

[1] Allais M.(1953):. Le comportement de l'homme rationnel devant le risque: critique des postulats et axiomes de l'école américaine, *Econometrica*, 21, 503-546.

[2] Benferhat S., Dubois D., Prade H. (1997): Nonmonotonic reasoning, conditional objects and possibility theory. *Artificial Intelligence*, 92, 259-276.

[3] Boutilier C. (1994): Toward a logic for qualitative decision theory, in *Proc. 4th. Inter. Conf. on Principles of Knowledge Representation and Reasoning (KR'94)*, Bonn, pp. 75-86.

[4] Bonet B. and Geffner H. (1996): Arguing for decisions: A qualitative model of decision making. *Proc. of the 12th Conf. on Uncertainty in Artificial Intelligence* (E. Horwitz, F. Jensen, eds.), Portland, OR, 98-105.

[5] Brafman R.I. and Tennenholtz M. (1997): On the axiomatization of qualitative decision criteria, in *Proc. 14th Nat. Conf. on A.I. (AAAI'97)*, pp. 76-81.

[6] Dubois D., Esteva F., Garcia P., Godo L., de Mantaras R. and Prade H. (1997): Fuzzy modelling of case-based reasoning and decision, in *Proceedings 2nd. Int. Conf. on Case Based Reasoning (ICCBR'97)*, (Leake & Plaza, eds.), Springer-Verlag. 599-611.

[7] Dubois D., Fargier H., Prade H. (1996): Refinements of the maximin approach to decision-making in fuzzy environment. *Fuzzy Sets & Systems*, 81: 103-122.

[8] Dubois D., Fargier H., Prade H. (1998): Choice under uncertainty with ordinal decision rules: a formal investigation. Research report 98-03 R, IRIT, Toulouse.

[9] Dubois D., Fodor J. C., Prade H. and Roubens M. (1996): Aggregation of decomposable measures with application to utility theory, in *Theory and Decision 41*, pp. 59-95.

[10] Dubois D. and Prade H. (1988): Possibility Theory. Plenum Press.

[11] Dubois D. and Prade H. (1995): Possibility theory as a basis for qualitative decision theory, in *Proc. of the 14th Int. Joint Conf. on Artificial Intelligence (IJCAI'95)*, Montreal, pp. 1924-1930.

[12] Dubois D., Prade H. and Sabbadin R. (1997): A possibilistic machinery for qualitative decision. *Working Notes of the AAAI'97 Spring Symposium. Series on Qualitative Preferences in Deliberation and Practical Reasoning*, Stanford, CA, March 24-26, pp. 47-54.

[13] Dubois D., Prade H. and Sabbadin R. (1997): Decision under qualitative uncertainty with Sugeno integrals — An axiomatic approach. *Proc. of the 7th Int. Fuzzy Systems Assoc. Cong. (IFSA'97)*, Prague, Czech Republic, Academia, Vol. I, 441-446.

[14] Ellsberg D. (1961): Risk, ambiguity and the Savage axioms. *Quarterly Journal of Economics*, vol. 75, p.643-669.

[15] Gilboa I. (1987) Expected utility with purely subjective non-additive probabilities, *Journal of Mathematical Economics*, 16, 65-88.

[16] Gilboa I. and Schmeidler D. (1995): Case-based Theory, *The Quarterly Journal of Economics*, pp. 607-639.

[17] Herstein I.N., Milnor J. (1953): An axiomatic approach to measurable utility, *Econometrica* 21, pp. 291-297.

[18] Lehmann D. (1996): Generalized qualitative probability: Savage revisited. in *Proc. of the 12h Inter. Conf. on Uncertainty in Artificial Intelligence*, (E. Horvitz and F. Jensen, eds.) pp. 381-388.

[19] Lehmann D. (1997): Non-standard utilities. Technical Report CS 97-15. Institute of Computer Science, The Hebrew University of Jerusalem.

[20] von Neumann J. and Morgenstern O. (1944): *Theory of Games and Economic Behaviour* (Princeton Univ. Press, Princeton, NJ).

[21] Pearl J. (1993): From qualitative utility to conditional "ought to", in *Proc. of the 9th Inter. Conf. on Uncertainty in Artificial Intelligence*, (D. Heckerman, H. Mamdani, eds.), pp. 12-20

[22] Sarin R., Wakker P. P. (1992): A simple axiomatization of nonadditive expected utility. *Econometrica*, vol. 60, No 6, 1255-1272.

[23] Savage L.J. (1972): The Foundations of Statistics. Dover, New York.

[24] Schmeidler D. (1989): Subjective probability and expected utility without additivity. *Econometrica*, 57, 571-587.

[25] Spohn W. (1988): Ordinal conditional functions: A dynamic theory of epistemic states. In Causation in Decision, Belief Change and Statistics, (W. Harper and B. Skyrms eds.), pp. 105-134.

[26] Sugeno M. (1977): Fuzzy measures and fuzzy integrals - A survey. In *Fuzzy Automata and Decision Processes*, (M.M. Gupta, G.N. Saridis and B.R. Gaines eds.), pp. 89-102, North-Holland.

[27] Tan S.W. and Pearl J. (1994): Qualitative decision theory. *Proc. of the 11th National Conf. on Artificial Intelligence (AAAI'94)*, Seattle, WA, 928-933.

[28] Whalen T. (1984): Decision making under uncertainty with various assumptions about available information. IEEE *Trans. on Systems, Man and Cybernetics*, 14, pp. 888-900.

[29] Yager R.R. (1979): Possibilistic decision making. IEEE *Trans. on Systems, Man and Cybernetics*, 9, pp. 388-392.

[30] Yager R.R. (1983): An introduction to applications of possibility theory. *Human Systems Management*, 2 pp. 246-269.

[31] Zadeh L.A. (1965): Fuzzy sets. *Information and Control* 8, pp. 338-353.

[32] Zadeh L.A. (1978): Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets & Systems*, 1, pp. 3-28.

Planning II

Encoding Planning Constraints into Partial Order Planning Domains

M. Baioletti, S. Marcugini, A. Milani
Dipartimento di Matematica ed Informatica
Università di Perugia
Via Vanvitelli
06100 Perugia - Italy
E-mail: {marco,gino,milani}@dipmat.unipg.it

Abstract

This paper presents a technique for encoding the representation of a constrained planning problem through the generation of an equivalent unconstrained planning problem. In many real situations the initial state/goals characterization of a planning problem is not satisfactory, this motivates the notion of a constrained planning problem, i.e. a planning problem for which the user specifies additional constraints on the problem solution in order to: give a limit to the length of the solution plan, use or avoid specific action instances, use particular strategies in adding steps or precedence constraints to the plan, reach some intermediate states. A common way to implement constrained planning is modifying the existing planner in order to take into account of the user additional constraints. In this paper we present a planner independent approach, which operates on the problem and constraints representation instead of modifying the planning algorithm. A planning constraint description language (PCDL) is therefore introduced. It is shown that a significant subset of PCDL can be encoded by modifying (domain preprocessing phase) the planning domain and goals. The original problem can then be solved by submitting the modified problem to the planner. The solutions to the modified problem defined by the new domain correspond (solution postprocessing phase) to the solutions of the original constrained problem. A significant result is that domain preprocessing phase has a linear time/space cost in the domain and additional constraints dimensions, and the solution postprocessing phase has also linear cost on the solution length. A theoretical consequence of the equivalence result is that representation of constraints and strategies does not require a planning model which is more powerful than an ordinary partial order planning.

1 Introduction

Planning problems have been widely characterized in terms of *end goals* or *goals of attainment*: end goals are specified by stating the conditions that should hold in the final state, after the execution of the plan [8]. Often solution plans that simply achieve end goals are unsatisfactory since real user have more complex goals. Examples of these *non end goals* or *extended goals* are: goals of *maintaining* (or *preventing* from) some conditions throughout plan execution (e.g. the power must be maintained on); goals of having *sequence of activities* in the plan (e.g. I want a plan for calling the bank prior to returning home); more complex attainment goals, such as requiring one or more *intermediate states* to be reached (e.g. a roundtrip goal from Florence to Rome requires the *intermediate* attainment goal of being in Florence); goal of having *temporal precedences* among activities or attainment goals (e.g. be in Florence, then go to restaurant); goals of having certain *codesignation constraints* among activities (e.g. I want to come back from Florence with the same car I used to go there). In addition the user can require that the final plan verifies some *strategies* or *efficiency criteria*. The solution should be reached *using (or avoiding) certain domain operators* (i.e. I want to take a plane but I do not want to build one) or the plan must verify conditions on plan length (e.g. prefer shorter plans), actions duration, plan cost or plan utility (e.g. obtain a solution that verify a minimum/maximum interval of duration/cost/utility).

The problem of specifying *non end goals* and building solution plans for them has been faced by different planning models based on languages able to express properties of sequences of state or events. In hierarchical task planning (HTN) [13], it is possible to specify activities goals (**do** goals) and decompositions (**perform** goals), while other models allow complex specification of activities (for instance see [17],[11],[9]).

UCPOP [18] allows the user to specify some other kinds of planning constraints, like axioms and safety constraints. An axiom is a domain law, i.e. an implication between facts, like "if a block A is on a block B, then B is above A". A safety constraint is a requirement that a given condition must not made true during the whole plan, like "don't close the door", although this condition is allowed to be true from the initial state, i.e. in our example the door can even be closed from the beginning, the important is that none must close it.

Unfortunately the implementations are scarce for some models, some other models sacrifice generality for efficiency (i.e. in the case of HTN) or have a different focus (i.e. interaction–coordination versus planning in multiagent models).

Our work uses a different approach by representing the *extended goals* inside the classical model, and considering all the *non end goals* as constraints on the final plan.

This point of view is rather different to other uses of constraints in planning, as in Tate [17], where plans are seen as constrained objects, or in Stefik [16], whose planning system, MOLGEN, is mainly based on the activity of constraints posting.

The main contribution of our work is showing that planning with constraints described in PCL-1, a language for defining planning constraints which will be introduced later, is equivalent to ordinary planning through a domain transformation. This result shows that instead of building up a new planning algorithm for solving a constrained problem, we can embed the problem constraints within a classical domain: from the constrained problem domain an equivalent classical problem domain is generated which contains only end goals.

The advantage is that the equivalent problem can be solved by a classical planner, making also possible to exploit the efficiency of certain recent fast implementations for the classical model (GRAPHPLAN [4], SATPLAN [10] etc.). Moreover since the approach is not bound to a particular planner implementation, new kind of constraints can be added without any modification in the planning algorithm.

In order to force the search space to produce plans which verifies the extended constraints, some dummy elements are introduced into the domain (i.e. dummy operators, initial facts and goals) and since the equivalent problem and its solutions will contain these dummy elements, dummy operators are to be removed in order to produce a solution to the original constrained problem through a solution decoding phase.

A remarkable result is that constraint coding and equivalent solution decoding are very efficient, being their complexity linear in time and space.

In the following section we introduce a planning constraint description language (PCDL) for specifying extended constraints.

In section 3 we characterize a significant subset of PCDL which can be embedded in classical domain. In section 4 the encoding rules are introduced. Expressivity, implementation issues and a discussion about efficiency of encoding/decoding phases conclude the paper.

2 Reference Planning Model and the constraints formalism PCDL

We shall refer to a very general planning formalism by which we describe a planning problem and its solutions.

A problem $\mathcal{P}$ is a triple $(\mathcal{I}, \mathcal{A}, \mathcal{G})$, where $\mathcal{I}$ is the description of the initial states, $\mathcal{G}$ is the goals set and $\mathcal{A}$ is the operators set.

An operator is characterized by a *name*, the lists of *parameters*, *preconditions* and *effects*.

A partial plan for a given problem $\mathcal{P} = (\mathcal{I}, \mathcal{A}, \mathcal{G})$, is a triple $(\mathcal{S}, \mathcal{T}, \mathcal{B})$ where $\mathcal{S}$ is the set of plan steps, that is instances of operators in $\mathcal{A}$. An instance is defined by a name of an existing operator a of $\mathcal{A}$ and a list of parameters, obtained by renaming the parameters of a with new names.

$\mathcal{T}$ is partial temporal order relation on the plan steps in $\mathcal{S}$, $\mathcal{B}$ is the codesignation/non codesignation relation on the variables present in the plan steps.

A solution Π is a partial plan $(\mathcal{S}, \mathcal{T}, \mathcal{B})$ such that it satisfies the **goals achievement** constraint, that is the goals have to be true in the final situation, and the **executability** constraint, that is each step s in $\mathcal{S}$ has to be executable (i.e. its preconditions are true) in the situation before its execution.

We extend the classical planning model by allowing the user to specify some planning constraints described with a reference language, called PCDL (for Planning Constraints Description Language).

A constrained planning problem $\mathcal{P}'$ is a quadruple $(\mathcal{I}, \mathcal{A}, \mathcal{G}, \mathcal{C})$, where $\mathcal{I}$, $\mathcal{G}$ and $\mathcal{A}$ have the same meaning as in the unconstrained case, and $\mathcal{C}$ is a PCDL constraint.

A solution Π for a constrained problem $\mathcal{P}$ is a partial plan such that Π solves the unconstrained version of $\mathcal{P}'$ and satisfies $\mathcal{C}$.

A constraint in PCDL is formulated by means of a formula, which can be either an atomic formula or a combination of formulas obtained using the following logical connectives and quantifiers: $(\forall s)\phi$, $(\exists s)\phi$, $\phi \wedge \phi'$, $\phi \vee \phi'$, $\phi \supset \phi'$, $\neg\phi$, $\#(s : \phi) \leq n$, $\#(s : \phi) \geq n$.
The quantifiers are meant to vary over the plan steps, n is an integer and ϕ, ϕ' are formulae.

$\#(s : \phi) \leq n$ is true if and only if the number of steps s for which ϕ holds does not exceed n and the meaning of $\#(s : \phi) \geq n$ is similar.

An atomic formula can be one of these predicates: $s = s'$, $s \prec s'$, trueAfter(s, p), $t = t'$, $t \in t'$ where s and s' are plan steps, $\preceq$ is the precedence relation between steps, o is an operator, p is a proposition and t and t' are terms.

We will use the usual notation $s \neq s'$ for $\neg(s = s')$ $t \neq t'$ for $\neg(t = t')$.

The predicate trueAfter(s, p) is true if and only if p is true after the execution of step s.

The terms are: name(s), value(s, v), parms(s), precond(s), effects(s).
Given a step s, the function name(s) returns the name of the operator of which s is an instance, and value(s, v) returns the value bound to the variable v of s.

parms(s), precond(s) and effects(s) return respectively the set of parameters, preconditions and effects of s.

PCDL is rich enough to express many useful constraints.

For instance a presence constraint, i.e. a given action a is to be used, like "use the action fly_to, i.e. fly at least once", can be expressed as

$$(\exists s)(\text{name}(s) = \text{fly\_to})$$

A safety constraint, i.e. some condition p must be true through the plan, like "do not close the door", can be expressed as

$$\neg(\exists s)(\text{door\_closed} \in \text{effects}(s))$$

A precedence constraint, i.e. some action a is to be executed before some other action b, like "knock the door before opening the door", can be expressed as

$$(\forall s)(\text{name}(s) = \text{open\_door} \supset (\exists s')(\text{name}(s') = \text{knock\_door} \wedge s' \prec s))$$

3 A Significant Subset of PCDL

We now restrict our investigations on a significant subset of the constraints for each of them we provide a semantics based on PCDL. All these constraints, as we will see later, have the nice feature that they can be *compiled* within the classical planning model.

The whole language formed by the following constraints, possibly combined by conjunction, will be called PCL-1.

3.1 Presence Constraints

The simplest forms of presence constraints are **AT-MOST-ONCE**(a) and **AT-LEAST-ONCE**(a). They mean that at most (least) one instance of the operator a can be present in S. The latter will be called *existence* constraint, the former *non duplicity* constraint.

Their formulations in PCDL are

$$\text{AT-MOST-ONCE}(a) \equiv (\forall s)(\forall s')((\text{name}(s) = a \wedge \text{name}(s') = a) \supset s = s')$$

and

$$\text{AT-LEAST-ONCE}(a) \equiv (\exists s)(\text{name}(s) = a).$$

The conjunction of both the previous constraints is the *uniqueness* constraint **EXACTLY-ONCE**(a) which requires the presence of a unique instance of a.

The constraints AT-LEAST-ONCE and AT-MOST-ONCE can be generalized in many ways. The most immediate are **AT-LEAST-TIMES**(n, a) and **AT-MOST-TIMES**(n, a), where n is an integer and a is any operator.

Their semantics is a straightforward extension of the previous constraints:

$$\text{AT-MOST-TIMES}(n, a) \equiv (\#(s : \text{name}(s) = a) \leq n)$$

and

$$\text{AT-LEAST-TIMES}(n, a) \equiv (\#(s : \text{name}(s) = a) \geq n)$$

Another generalization are given by **AT-LEAST-ONCE**(A) and **AT-MOST-ONCE**(A), where A is a subset of the whole operators set.

Their semantics is simple: s verifies this constraint if it contains at most (at least) one instance of one operator belonging to A, that is

$$\text{AT-MOST-ONCE}(A) \equiv (\forall s)(\forall s')((\text{name}(s) \in A \land \text{name}(s') \in A) \supset s = s')$$

and

$$\text{AT-LEAST-ONCE}(A) \equiv (\exists s)(\text{name}(s) \in A).$$

There exist many ways to combine both possibilities (n instances and m operators), but only few of them are really interesting.

The couple of contraints **AT-LEAST-AMONG**(n, A) and **AT-MOST-AMONG**(n, A) mean that s has to contain at least (at most) n instances of operators belonging to A, taking into account of multiple instances of the same operator.

In PCDL they are

$$\text{AT-MOST-AMONG}(n, A) \equiv (\#(s : \text{name}(s) \in A) \leq n)$$

and

$$\text{AT-LEAST-AMONG}(n, A) \equiv (\#(s : \text{name}(s) \in A) \geq n)$$

A very useful constraints is **AT-MOST-AMONG** when A coincides with the set of all operators.

In this way it is possible to express that any solution must not have more than n steps, that is the user can select an upper bound to the plan length.

A particular generalization of these constraints is obtained by allowing to give a different integral weigth to each operator.

Weigths could be interpreted as actions costs, utility or durations and the upper (lower) bound n would be the maximum/minimum plan cost–utility–duration allowed.

Another combination is the constraint **ONE-OF**(A), where A is always a subset of the operators set.

This constraint is satisfied by s if and only if s has only instances of only one operator taken from A. In other word **ONE-OF** expresses a sort of mutual exclusivity on a set of operators: if one of them is used as a plan step, no other operator can be instanciated any more.

In PCDL ONE-OF(A) would be expressed as

$$\text{ONE-OF}(A) \equiv (\exists s)(\text{name}(s) \in A) \supset (\forall s')(\text{name}(s') = \text{name}(s) \lor \neg\text{name}(s') \in A)$$

If the user wants to express a stronger mutual exclusion, extended also to the instances of the same operator, then he or she must use AT-MOST-AMONG($1, A$).

3.2 Absence Constraints

An absence constraint **EXCLUDE**(a) means that a particular operator a must not be present in the solution. In PCDL it is

$$\text{EXCLUDE}(a) \equiv \neg(\exists s)(\text{name}(s) = a)$$

3.3 Order Constraints

The order constraints are **BEFORE**(a, b) and its opposite **AFTER**(a, b). They impose that each instance of a has to be executed before (after) each instance of b.

In PCDL, for instance, BEFORE would be described as

$$\text{BEFORE}(a, b) \equiv (\forall s)(\forall s')((\text{name}(s) = a \land \text{name}(s') = b) \supset s \prec s').$$

3.4 Achievement Constraints

The achievement constraint **ACHIEVE**(p), where p is a proposition, states that at some time the condition p must be achieved.

In PCDL it would be rendered as

$$\text{ACHIEVE}(p) \equiv (\exists s)(p \in \text{effects}(s)).$$

Since the simple request for achieving a given condition could be easily satisfied by specifying a final goal, it makes sense only to use this kind of constraints in conjunction with order constraints.

We therefore allow to add a unique label to each *ACHIEVE* constraint, making possible to express an order relation among the time instants in which the conditions are to be achieved. These order constraints are defined by using BEFORE (or AFTER) between the labels attached to the corresponding ACHIEVE constraints.

For instance "be in Rome, then be in Florence" would be stated by the constraints
L1:ACHIEVE(be-in-Rome)
L2:ACHIEVE(be-in-Florence)
BEFORE(L1,L2)
The equivalent constraint in PCDL would be

$$(\exists s)(\exists s')(\text{trueAfter}(s, \text{be-in-Rome}) \land \text{trueAfter}(s', \text{be-in-Florence}) \land s \prec s')$$

In this way it is possible to define, by means a planning constraint, a construct equivalent to the *achieve* tasks present in HTN planning [13].

In a very similar way it would be also possible to provide a construct similar to the *do* task present in HTN planning [13], leading to the possibility of defining a complete task network, exactly as in [2].

3.5 Codesignation Constraints

A codesignation constraint is **CODES**$(x : a, y : b)$, where a, b are operators, x is a parameter of a and y is a parameter of b. It means that the parameters x and y must be bound to the same value.

In PCDL it would be rendered as

$$(\forall s)(\forall s')(\text{name}(s) = a \land \text{name}(s') = b \supset \text{value}(s, x) = \text{value}(s', y))$$

The opposite is the non codesignation constraint **NON-CODES**$(x : a, y : b)$, where a, b, x, y are like the previous constraint. It means that the parameters x and y must be bound to the different value.

3.6 Conditional Constraints

In this class, constraints are active only when a certain condition is true.

We have the *conditional presence* constraints **NEED-AFTER**(a, b) and its opposite **NEED-BEFORE**(a, b). They means that if an instance s of operator a is present then there must exist an instance s' of operator b such that s precedes (follows) s'. In terms of PCDL they are expressed as

$$\text{NEED-AFTER}(a, b) \equiv (\forall s)(\text{name}(s) = a \supset (\exists s')(\text{name}(s') = b \land s \prec s'))$$

and as

$$\text{NEED-BEFORE}(a, b) \equiv (\forall s)(\text{name}(s) = a \supset (\exists s')(\text{name}(s') = b \land s' \prec s))$$

Another kind of conditional constraint is the *triggered action* **IF**(p, a). It states that when p becomes true, then a has to be executed.

In PCDL this constraint is

$$\text{IF}(p, a) \equiv (\forall s)(\text{trueAfter}(s, p) \supset (\exists s')(\text{name}(s') = a \land s \prec s'))$$

4 Compiling constraints in the domain

In this section we will show how it is possible to find an appropriate trasformation to be applied in the domain such that the new planning problem has only solutions which respect the desired constraints.

In other words with these transformations we are able to restate a problem inglobing the additional constraints within the domain itself.

A precompilation phase producing a new unconstrained planning problem from a given constrained problem is defined using the methods described above.

These techniques add some dummy facts to the initial states and/or the goals and/or the preconditions or effects of some existing operators. They can sometimes add some new dummy operators to the domain.

The presence of dummy items causes the solution plan to be sometimes expressed in a language richer than the original planning language. Hence a postcompilation phase is needed to translate the solution in the old language, i.e. by deleting dummy operators.

In the following we expose for each type of constraints defined in the previous section its own method of compilation. We suppose to be able to generate dummy items by means of a system function like Common Lisp GENSYM, we are therefore guaranteed that these symbols will be new (not already present in the domain) and it will be not used any more for the compilation of other constraints.

Each method will be described in terms of what it adds to the domain, which will always be denoted by an initial state $\mathcal{I}$, a set of user goals $\mathcal{G}$ and a set of operator $\mathcal{A}$.

The presence constraint **AT-LEAST-ONCE**(a) is translated by inserting a new dummy goal u_a into the user goals $\mathcal{G}$ and adding it to the effects of a.

Since every solution s must satisfies the goal u_a, which is asserted only by a, the presence of an instance of a is guaranteed.

Dually, the presence constraint **AT-MOST-ONCE**(a) is translated by inserting a new dummy fact u_a into the initial state $\mathcal{I}$ and adding it to the preconditions of a, while its negated form $\neg u_a$ is added to the effects of a.
It is easy to see that no more than one instance s of a can exist in the plan, since s deletes the precondition of a and there are no other step which can reassert

it. It is worth noting that this constraint does not require that any instance of a will ever exist.

The compilation of the constraint **EXACTLY-ONCE**(a), as a conjunction of both the previous constraints, can be obtained using the *additivity principle*, which will be stated later. However this is more concise way to compile it: the same dummy fact u_a is added to the initial state and the preconditions of a, while $\neg u_a$ is added to its effects and to the goals $\mathcal{G}$.

The constraint **AT-LEAST-TIMES**(n, a) is compiled by iterating (in some way) the compilation of the constraint AT-MOST-ONCE, that is by creating a new dummy fact u_a, adding the n dummy facts $u_a(1), \ldots, u_a(n)$ to the goals and the generic $u_a(x)$ to the effects of a.

Satisfying n different goal $u_a(1), \ldots, u_a(n)$ requires at least n instances of a (which is the only operator asserting $u_a(x)$), therefore the constraint is automatically verified.

The constraint **AT-MOST-TIMES**(n, a) is translated in a very similar way by creating a new dummy fact u_a, adding the n dummy facts $u_a(1), \ldots, u_a(n)$ to I, the generic fact $u_a(x)$ to the preconditions of a and $\neg u_a(x)$ to the effects.

The translation of **AT-MOST-ONCE**(A) and **AT-LEAST-ONCE**(A) does not present any difference with respect to the related previous constraints: the same appropriate transformation has to performed to every operator belonging to A.

AT-LEAST-AMONG and **AT-MOST-AMONG** are translated in a very similar way with respect to **AT-LEAST-TIMES** and **AT-MOST-TIMES**.

The possibility of adding an integral weight w to an operator a is handled by adding w different preconditions and/or effects $u_a(x)$ (or its negation) to a.

The constraint **ONE-OF**(A), where $A = \{a_1, \ldots, a_m\}$, is compiled by creating a new dummy fact o_A, inserting in $\mathcal{I}$ the m dummy facts $o_A(1), \ldots, o_A(m)$, adding to preconditions of each operator a_i the new fact

$$\bigwedge_{j \neq i} o_A(j)$$

and finally adding the fact $\neg o_A(i)$ to the effects of a_i. When there is no instance of any operator belonging to A, then potentially any of them can be added, but as one of these operators a_i is selected then the presence of instances of any other operator a_j is forbidden, because a_i negates a precondition of a_j (which cannot be re-asserted), while it is still possible to have other instances of the same operator a_i.

The translation of a **BEFORE** constraint is accomplished by creating a new dummy fact p_{ab}, by adding it to $\mathcal{I}$ and to the preconditions of b, and by adding its negation to the effects of b.

A constraint **NEED-BEFORE**(a, b) is compiled by adding a dummy fact n_{ab} to the preconditions of a and to the effects of b.

NEED-AFTER(a, b) is compiled by creating a new fact n_{ab}, by adding it to $\mathcal{I}$, to $\mathcal{G}$ and to the effects of b, and by adding $\neg n_{ab}$ to the effects of a.

A constraint **ACHIEVE**(p) is compiled by adding to the domain a new operator a_p, whose preconditions are only p and effects are empty.

This operator is then modified as if there was an implicit uniqueness constraint in order to guarantee to execute it once. Its execution would require the achievement of the desired condition p.

Possible order relations among other achieve constraints must be handled in the same way as a **BEFORE** constraint.

The constraints **CODES**$(x : a, y : b)$ and **NON-CODES**$(x : a, y : b)$ are translated in this way: two new predicates e_a and e_b are created, $e_a(x)$ is added to the effects of a, $e_b(y)$ is added to the effects of b, then a new operator EQ is created with preconditions $e_a(x) \wedge e_b(y) \wedge x = y$ (for CODES) or $e_a(x) \wedge e_b(y) \wedge x \neq y$ (for NON-CODES) and no effects.

A special conditions must be added in order to execute the EQ (which imposes the required constraint) only when a and b are both present in the plan.

This is equivalent to a composite **NEED-AFTER**$(a \wedge b, OP)$ and can be accomplished with two other new dummy facts p_a and p_b, their conjunction is added to $\mathcal{I}$, their disjunction is added to $\mathcal{G}$, a will negate p_a and b will negate p_b.

It is worth noticing that this compilation is possible only when the planner allows for disjunctive goals and preconditions.

The compilation of the constraint **IF**(p, a) is somehow complex. The simplest case is when p is atomic. The constraint is translated by creating a new dummy fact u, adding it to $\mathcal{I}$ and $\mathcal{G}$, and modifying those operators o which have p among their effects. Each o takes as a new effect $\neg u$. Finally u is added to the effect of operator a.

If some operator adds a fact q which unifies with p, then the new effect $\neg u$ must be conditioned to the most general unificant between p and q. For instance if p is at(Rome) and o has an effect at(X) then the new conditional effect *WHEN X=Rome THEN* $\neg u$ should be added.

When p is a disjunction of several propositions, then the obvious relation

$$\text{IF}(p \vee p', a) \equiv \text{IF}(p, a) \wedge \text{IF}(p', a)$$

holds, therefore the additivity principle can be used to compile the constraint.

A slightly more difficult situation appears when p is a conjunction of several propositions. For instance if $p = p_1 \wedge p_2$ then u_1 and u_2 have to be created, their conjunction added to $\mathcal{I}$ and to the effects of a and their disjunction has to included in $\mathcal{G}$.

The modification of the operators should change in this way: if an operator o has effect p_1 then add a new effect $\neg u_1$, while $\neg u_2$ concerns to those operators which add p_2.

Finally the **EXCLUDE** constraint can be translated by trivially deleting the corresponding operators or by adding to them a non existing precondition.

4.1 Compiling conjunctive constraints

When the user expresses a conjunction of constraints, that is constraints which must be achieved at the same time, it is possible to use a technique of compilation that we called *additivity principle.*

It states that if a constraint C is expressable as a conjunction of several constraints $C_1, \ldots, C_n$ (which must obviously not be in contradiction) then C is compiled by sequentially compiling (in some order) each single constraint C_i.

In a more formal way if we denote by *compile* the function that given a problem P and a constraint C returns the new problem P' following the rules described through the previous section, then

$$\text{compile}(P, C_1 \wedge C_2 \ldots \wedge C_n) = \text{compile}(\text{compile}(\ldots \text{compile}(P, C_1), C_2), \ldots, C_n)$$

This principle can be justified by observing that each method of translation makes no hypothesis on the problem P it has to modify: it just sees which is the kind of the constraint C is to be translated and applies some fixed rule to P. Therefore P can be also the result of a previous compilation of another constraint.

For what it concerns disjunction, it is fairly clear that it is impossible to compile in a disjunctive way, i.e. there is no simple way to obtain the compilation of a constraint like EXCLUDE(a) $\vee$ BEFORE(a, b), starting from the compilations of the single disjuncts.

Therefore disjunction, when possible, has to be handled in a separate way, depending on the basic constraints.

5 Expressivity, complexity and implementation issues

The above described procedure COMPILE thus transforms a constrained planning problem $\mathcal{P}'$ into an unconstrained version $\mathcal{P}$, embedding $\mathcal{C}$ in the domain.

An obvious translation function ω from the space of plans of $\mathcal{P}$ into the space of plans of $\mathcal{P}'$ is needed in order to express a solution of $\mathcal{P}$ in the same language as the solutions of $\mathcal{P}'$, because for translating some constraints a dummy operator is added.

Therefore ω simply neglects these dummy operators to the domain of $\mathcal{P}'$.

In the following we will denote by $\ll \mathcal{P} \gg$ and $\ll \mathcal{P}' \gg$ respectively the space of solution for $\mathcal{P}$ and $\mathcal{P}'$ showing that these two set are equivalent.

Since each method of compilation produces an unconstrained problem whose solutions are forced to respect the constraints and solve $\mathcal{P}'$ in the ordinary meaning, it is easy to see that for any solution s of $\mathcal{P}$, $\omega(s)$ solves the original constrained problem $\mathcal{P}'$, that is

$$\forall s \in \ll \mathcal{P} \gg \; \omega(s) \in \ll \mathcal{P}' \gg$$

On the other hand, for each solution s' of $\mathcal{P}'$ it is possible to construct an equivalent solution s of $\mathcal{P}$, by reverting, in some way, the effect of ω, i.e. by adding the required dummy items to s'. Therefore we get that

$$\forall s' \in \ll \mathcal{P}' \gg \; \exists s \in \ll \mathcal{P} \gg \; \omega(s) = s'$$

Combining these two properties we obtain that $\ll \mathcal{P}' \gg$ is equivalent to $\ll \mathcal{P} \gg$ modulo ω, i.e.

$$\ll \mathcal{P}' \gg = \{\omega(s) : s \in \ll \mathcal{P} \gg\}.$$

These features have a strong impact in expressivity. Since it is clear that also a usual unconstrained planning problem can be stated in PCL-1 we can conclude that *our PCL-1 planning model has the same expressive power as the ordinary planning model.*

Our concept of *same expressivity power* is similar to Baader's [1] expressivity language definition, used also in [7] for showing that HTN planning is more expressive than ordinary planning.

This result is also strongly related to the proof of equivalence between TN and ordinary planning (see [2]); moreover a TN problem planning, as we have seen before, can be stated in terms of PCL-1, therefore TN planning is also equivalent to planning with PCL-1.

The expressivity equivalence moreover shows that some particular kind of constraints, like plan length, actions costs, mutual exclusion relations, which are usually thought as *strategy rules*, can be embedded directly in the domain.

The equivalence result does not mean the constraints are useless, because even if it is possible to encode directly them in the domain, the user surely prefers to describe constraints separately from the domain and to use a preprocessor that performs the encoding.

From the efficiency point of view it should be noted that the amount of dummy items added to the domain is, for each method of compilation, linear in the number and dimension of constraints. Also the times needed to the precompilation and postcompilation phases are linear according the size of their input data.

We tried our approach by implementing a preprocessor for the target planner UCPOP [18]. It takes a constrained planning problem expressed in PCL-1 adapted to action/fact description of UCPOP and produces its equivalent unconstrained version, which can be solved by UCPOP.

The main feature of the precompilation method is that we are able to solve a constrained problem without the necessity of writing a constrained planner, but by just using an existing ordinary planner.

Acknowledgements

This work has been partially supported by Special Project "Strategie di controllo innovative per sistemi di Intelligenza Artificiale (SCI*SIA)" of Italian National Research Council–C.N.R.

References

[1] F. Baader. *A Formal Definition for Expressive Power of Knowledge Representation.* In Proceedings of European Congress of Artificial Iintelligence, 1990.

[2] M. Baioletti, S. Marcugini, A. Milani. *Task Planning and Partial Order Planning: A Domain Transformation Approach.* In Proceedings of European Congress on Planning, 1997.

[3] A. Barrett, D. Weld. *Schema Parsing: Hierarchical Planning for Expressive Language.* In Proceedings of National Conference of Artificial Intelligence, 1994.

[4] A.L. Blum, M.L. Furst. *Fast Planning through Planning Graph Analysis.* Artificial Intelligence 90(1-2): 281-300, 1997

[5] D. Chapman. *Planning for conjunctive goals.* Artificial Intelligence 32, 1987

[6] K. Erol, J. Hendler, D.S. Nau. *Semantics for Hierarchical Task-Network Planning.* Tech. rep. CS-TR-3239, Univ. of Maryland, March 1994

[7] K. Erol, J. Hendler, D.S. Nau. *HTN Planning: Complexity and Expressivity.* In Proceedings of National Conference of Artificial Intelligence, 1994.

[8] M. Georgeff *Planning.* In Annual Review of Computer Science, vol. 2, 1987

[9] M. Georgeff *A theory of action for multiagent planning.* In Proceedings of National Conference of Artificial Intelligence, 1984.

[10] H. Kautz, B. Selman *Pushing the envelope: Planning, propositional logic, and stochastic search.* In Proceedings of National Conference of Artificial Intelligence, 1996.

[11] A.L. Lansky *A representation of parallel activity based on events, structure, and causality.* In Reasoning about Actions and Plans: Proceedings of the 1986 Workshop, 123-159.

[12] E.D.P. Pednault. *Synthesizing plans that contains actions with context-dependent effects.* Computational Intelligence, Vol. 4, 1988.

[13] S. Kambhampati. *A Comparative Analysis of Partial Order Planning and Task Reduction Planning*, SIGART Bull. 1995.

[14] S. Kambhampati, J. Hendler. *A validation structure based theory of plan modification and reuse.* Artificial Intelligence, May 1992.

[15] E.D. Sacerdoti, *The nonlinear nature of plans.* In Proceedings of International Joint Conference of Artificial Intelligence, 1975.

[16] M. Stefik. *Planning with constraints (MOLGEN: Part 1)* in Artif. Intell. 16(2): 111-140, 1981.

[17] A. Tate. *Representing Plans as a Set of Constraints – the <I-N-OVA> Model.* In Proceedings of Conference on Artificial Intelligence Planning Systems, 1996.

[18] J.S. Penberthy and D. Weld. *UCPOP: A Sound, Complete Partial Order Planner for ADL.* In Proceedings of Third International Conference of Principles of Knowledge Representation and Reasoning, 1992.

[19] Q. Yang. *Formalizing Planning Knowledge for hierarchical planning,* Computational Intelligence, Vol 6, pag. 12-24, 1990

A Planning Algorithm not based on Directional Search

Jussi Rintanen
Universität Ulm
Fakultät für Informatik
Albert-Einstein-Allee
D-89069 Ulm, Germany

Abstract

The initiative in STRIPS planning has recently been taken by work on propositional satisfiability. Best current planners, like Graphplan, and earlier planners originating in the partial-order or refinement planning community have proved in many cases to be inferior to general-purpose satisfiability algorithms in solving planning problems. However, no explanation of the success of programs like Walksat or relsat in planning has been offered. In this paper we discuss a simple planning algorithm that reconstructs the planner in the background of the SAT/CSP approach.

1 INTRODUCTION

Many of the recent interesting results in AI planning did not originate in traditional planning research, but in work on algorithms for checking the satisfiability of propositional formulae. STRIPS planning problems have been used as benchmarks to test SAT algorithms based on greedy local search [Kautz and Selman, 1992; Kautz and Selman, 1996], and new developments [Bayardo, Jr. and Schrag, 1997] of the well-known Davis-Putnam procedure. As a contribution to planning research, these SAT algorithms have proved to be in many cases orders of magnitude faster planners [Kautz and Selman, 1996] than algorithms specifically designed for planning, thereby pointing out the possibility of dramatic improvements and new interesting lines of research.

Planning by satisfiability was first investigated by Kautz and Selman [1992] by means of a stochastic search algorithm. In this approach, problem instances are translated to sets of propositional formulae that include frame axioms, formulae describing the effects the operators have, and facts that hold in the initial and goal state. The satisfiability algorithm finds an assignment of truth-values to the propositions, and a plan is obtained from the propositions that correspond to operator applications. Interestingly, there are several ways to express the frame axioms and to make the set of formulae more concise, which has led to a thread of research on the quantitative properties of different encodings [Kautz and Selman, 1996; Kautz *et al.*, 1996; Ernst *et al.*, 1997].

Parallel to advances in solving satisfiability problems by stochastic search, improvements to the well-known Davis-Putnam procedure [Davis *et al.*, 1962] have been discovered. State-of-the-art implementations include Crawford's [1996] tableau and Freeman's [1995] Posit. Techniques that have made it possible to solve planning problems of the same difficulty with the Davis-Putnam procedure as with *Walksat* [Kautz and Selman, 1996] include lookback techniques from constraint satisfaction research [Bayardo, Jr. and Schrag, 1997] and heuristics based on unit resolution [Li and Anbulagan, 1997].

In this paper, we directly apply techniques from satisfiability algorithms to STRIPS planning, bypassing the route via translations to SAT. This paper, however, is not about satisfiability testing, as the algorithm we present can be and should be completely understood in terms of planning per se. It is important to devise this algorithm for several reasons. The functioning of SAT algorithms when they are solving planning problems has not been analyzed in earlier research, which has kept the reason for the success of these algorithms from being widely known. For example, the role of various details of problem encodings, like the use of invariants, has been given only a quantitative explanation as a factor that improves the runtimes of SAT algorithms. Furthermore, it is important to identify the boundary between satisfiability testing and planning. We believe that there are a lot of techniques that can improve the performance of planning algorithms, but that are not likely to be discovered by research on propositional satisfiability. To find the boundary between the two fields and to identify lines of research relevant to planning, it is necessary to understand what a SAT algorithm is able to do when it is trying to find a plan.

2 WORKINGS OF A SAT-BASED PLANNER

In this section we illustrate what takes place when satisfiability algorithms like the Davis-Putnam procedure solve a planning problem represented as propositional formulae. The representation of planning as satisfiability problems [Kautz and Selman, 1992; Kautz *et al.*, 1996] consists of the frame axioms, axioms describing the preconditions and effects of operators, and formulae describing the initial state and the goal. We take as an example the axiomatization from [Kautz *et al.*, 1996] with explanatory frame axioms [Haas, 1987; Schubert, 1990] and parallel operations.

For a fluent l that can be made true by operators $o_1, \ldots, o_n$, explanatory frame axioms are

$$(\overline{l^t} \wedge l^{t+1}) \rightarrow (o_1^t \vee \cdots \vee o_n^t) \qquad (1)$$

where $t \in \{0, \ldots, f\}$ is an integer referring to a point of time and f is the plan length. An operator application implies its preconditions and postconditions, that is, if the preconditions of o are $l_1, \ldots, l_n$ and the postconditions $l'_1, \ldots, l'_{n'}$, then we have the axiom

$$o^t \rightarrow (l_1^t \wedge \cdots \wedge l_n^t \wedge l_1'^{t+1} \wedge \cdots \wedge l_{n'}'^{t+1}). \qquad (2)$$

Fluents $l_1, \ldots, l_n$ true at the initial state are encoded simply as the unit clauses $l_1^0, \ldots, l_n^0$, and the goal fluents $g_1, \ldots, g_m$ as the unit clauses $g_1^f, \ldots, g_m^f$. Finally, there are axioms that prevent the simultaneous application of operators that are mutually dependent.

Most of the inferences in the Davis-Putnam procedure when it is solving a planning problem are based on unit resolution and unit subsumption. From a unit clause o^t the preconditions at t and postconditions at $t+1$ are obtained. An important pattern of inference is started by unit clauses $\neg o^t$. First, clauses from axioms in Eq. 2 are deleted by unit subsumption. Second, and more importantly, unit resolution together with frame axioms (Eq. 1) may force the truth-value of a fluent to persist across a step of time or to force the application of an operator. If $\neg o^t$ was the last literal in the consequent we have $l^t \vee \overline{l^{t+1}}$, that is, if the fluent l is false at t then it is false also at $t+1$. Depending on the presence of $\overline{l^t}$ or l^{t+1}, this may yield $\overline{l^{t+1}}$ or l^t. If o^t was next to the last literal in the consequent and we already had $\overline{l^t}$ and l^{t+1}, unit resolution forces the application of the remaining operator o' at t. Similarly, a unit clause representing a fluent together with a frame axiom 1 may propagate its truth-value one step backward or forward or force an operator application.

The above inference patterns are very powerful in making forward inferences from the initial state. As the initial state in STRIPS planning determines the truth-values of all fluents, unit resolution directly lets us conclude which operators are not applicable at time 0 because their preconditions are false. For many fluents this – like shown above – allows to infer their persistence, which in turn yields the inapplicability of other operators at time 1, and so on.

After unit simplifications, the Davis-Putnam procedure does a case analysis on a literal l, in one case adding l to the clause set and in the other $\overline{l}$, in both cases enabling further inferences by unit simplifications.

The algorithms by Kautz and Selman [1992; 1996] based on greedy local search work quite unlike the Davis-Putnam procedure. These algorithms start from a randomly chosen truth-assignment that is repeatedly modified by reversing truth-values of literals so that more clauses are satisfied. However, flips roughly corresponding to unit simplifications are likely to be made by *Walksat* as they increase the number of satisfied clauses.

3 THE ALGORITHM

In this section we give a new algorithm for STRIPS planning. The algorithm is similar in structure to the Davis-Putnam procedure for propositional satisfiability. Each recursive call consists of applying a set of efficient inference rules, followed by a case analysis. The main procedure of the algorithm, given in Figure 3 uses a plan length as a parameter, and it is iteratively called for all plan lengths starting from 0 until a solution is found. Currently we employ no method for detecting insolvability, so the planner runs forever on problem instances that have no solution. The proofs of soundness and completeness of the algorithm are straightforward.

A distinguishing characteristic of the algorithm is that – unlike most earlier planning algorithms – the search does not start from the goal state and proceed towards the initial state. Instead, operator applications at any points of time may be chosen for the case analysis that makes the search tree branch. In one of the subtrees the operator is applied, and in the other it is not. This is in strong contrast with Graphplan [Blum and Furst, 1995] that generates a branch for every minimal set of operators that produces the current subgoal, and with algorithms like SNLP [McAllester and Rosenblitt, 1991] that branch on possible ways of extending a partially ordered plan so that a threat is removed or an operation in the incomplete plan has its precondition fulfilled.

An inference technique from satisfiability testing we have found useful is *failed literal detection*, proposed by Freeman [1995] and investigated by Li and Anbulagan [1997]. We discuss it in the section on invariants where we show that it enables backward inferences from the goal state.

```
procedure plan()
  allocate space for local arrays op' and prop';
  if applyrules() then return false; end if
  if op[t,o] = Unknown for no t,o then return true; end if
failedLiteralDetection:
  foreach operator o and time t such that op[t,o] = Unknown do
    op' := op; prop' := prop;
    op[t,o] := True;
    if applyrules() then (* Contradiction was derived. *)
      op := op'; prop := prop';
      op[t,o] := False;
      if applyrules() then return false;
    else
      if score for op,prop improves the previous best score
      then t1 := t; o1 := o; p1 := 1; end if
      op := op'; prop := prop';
      op[t,o] := False;
      if applyrules() then (* Contradiction was derived. *)
        op := op'; prop := prop';
        op[t,o] := True;
        applyrules();
      else
        if score for op,prop improves the previous best score
        then t1 := t; o1 := o; p1 := 0; end if
        op := op'; prop := prop';
      end if
    end if
  end foreach
  if operator applications were derived
  then goto failedLiteralDetection end if;
  op' := op; prop' := prop;
  if p1 = 1 then op[t1,o1] := True else op[t1,o1] := False; end if
  if plan() then return true; end if
  op := op'; prop := prop';
  if p1 = 1 then op[t1,o1] := False else op[t1,o1] := True; end if
  return plan();
end
```

Figure 1: The planning algorithm

This technique is also the basis for selecting an operator for branching. Failed literal detection proceeds by attempting to derive literals by proof-by-contradiction: assume the literal is true, and if contradiction can be derived by unit simplifications, conclude the negation of the literal, otherwise estimate the usefulness of branching on the literal in terms of the number of short clauses it yielded. In our case, we assume that a certain operator is or is not applied at a certain moment of time, and for branching we select the operator (in)application that reduces the number of unknown operation applications and proposition values most.

The main procedure of the algorithm is given in Figure 3. Before calling it, the elements of arrays `op` and `prop` are initialized to Unknown, `prop[0,p]` for all p is assigned the truth-value of proposition p in the initial state, and similarly for `prop[len,p]` and the goal state, where `len` is the plan length currently considered. If the procedure returns true, a plan can be read from the array `op`.

The function `applyrules` performs the following inferences and returns *true* if a contradiction was detected, that is, an assignment of True to an element of `op` or `prop` that had value False was attempted, or vice versa.

1. If operator o is assigned true at t, then make the preconditions of o true at t and the postconditions true at $t+1$.
2. If all operators with l in the postcondition are assigned false at t, then do the following. If l is false at t then make l false at $t+1$. If l is true at $t+1$ then make l true at t.
3. If operator o is assigned true at t, assign false at t to all operators that make the precondition of o false, and assign false at t to all operators the precondition of which is made false by o.

The second rule encodes the assumption that only operations change the value of propositions. The third rule makes the planner a partial-order planner: operators that can be executed in any order without affecting the outcome may be executed simultaneously.

Additional rules are used for speeding up inferences – especially to reduce the need for failed literal detection which is expensive – even though they are not required for the correctness of the algorithm. For example, if a proposition changes truth-value at time t and all but one operator making the change have been chosen to be not applied, the remaining operator has to be applied. And if the precondition is false at t or a postcondition is false at $t+1$, then the operator is not applied at t. It is easy to verify that these inferences correspond to the ones sanctioned by unit resolution in planners based on SAT, for example with the encoding in [Kautz *et al.*, 1996] with parallel operations and explanatory frame axioms.

Note that like the Davis-Putnam procedure, as long as case analysis is not needed, the algorithm runs in polynomial time.

As an alternative to doing failed literal detection and branching on operator applications, it is also possible to do it on propositions, or both operator applications and propositions. What is the best alternative depends on the properties of the class of planning problems to be solved. Doing failed literal detection on both operator applications and propositions of course prunes the search space at least as much as the other alternatives, but it is more expensive.

4 INVARIANTS

The basic algorithm given in the previous section sanctions most of the desired inferences from the initial state. For example in the blocks world domain, if there is a stack of n

blocks on top of block A, we can infer that after n moves A is still in its initial position. We would like to make similar inferences starting from the goal state. For example, if A is on top of B and B is on top of C at time t, then B is on top of C at $t-1$. To make this inference, we need to show that it is not possible to move B on top of C at $t-1$. This inference uses failed literal detection. Assume that B is moved on top of C at $t-1$. The precondition of this operation is that B and C are clear at $t-1$. The moving of A on B cannot be done in parallel with the move of B on C because the former makes the precondition of the latter false. Hence A cannot be moved on B at $t-1$. Therefore A is on B at $t-1$. But then B cannot be clear at $t-1$, which contradicts the precondition of moving B on C.

What is missing in the algorithm in the previous section is the last step that whenever A is on B, B cannot be clear. This is a fact that is not explicit in the facts that A is on B and B is on C. The goal state and many intermediate states are incomplete in that they do not specify the truth-values of all propositions. To extend the state descriptions we need *invariants* derived from the operators and the initial state[1]. In the above example the relevant invariant says that either B is not clear or A is not on B. Most of the work on SAT-based planning has used this kind of invariants to speed up plan search [Kautz and Selman, 1992; Kautz and Selman, 1996]. Invariants (usually incompletely) characterize the set of states reachable from the initial state.

The invariants used in the algorithm consist of 2-literal clauses, which is the relevant form of invariants in most of the benchmark domains, like the blocks world, the rocket domain, and the logistics domain. In the blocks world the only invariants that cannot be represented as 2-literal clauses are those that state that the *on* relation is acyclic and if a block is not clear then some block has to be on top of it. Invariants are incorporated in the algorithm as the following rule: if $l \vee m$ is an invariant and l becomes False at t, then make m True at t.

4.1 COMPUTATION OF INVARIANTS

We have devised an algorithm for computing 2-literal invariants. Its structure is similar to the invariant algorithms in computer-aided verification [Bensalem *et al.*, 1996]. The algorithm starts with a candidate invariant Q', consisting of all the 2-literal clauses true in the initial state, and weakening it repeatedly by removing clauses that are made false by an operator application. This is formalized as the computation of fixpoints of monotonic functions R_O. The definition of R_O for a set of operators $O = \{o_1, \ldots, o_n\}$ is as follows.

$$
\begin{aligned}
R_O(V) &= F_{o_1}(F_{o_2}(\cdots F_{o_n}(V)\cdots)) \\
F_{\langle p,e\rangle}(V) &= \begin{cases} V, \text{ if } V \cup p \models \bot, \text{ and otherwise} \\ \{a \vee b \in V | \neg a \notin e \text{ or } b \in U(V,p,e), \\ \qquad \neg b \notin e \text{ or } a \in U(V,p,e)\} \end{cases} \\
U(V,p,e) &= \{l \in L | V \cup p \models l\} \backslash \{\bar{l} | l \in e\} \cup e
\end{aligned}
$$

The function F_o takes a set of clauses, and deletes the ones the truth of which the operator o does not preserve. The function U performs an update. It computes the set of literals $U(V,p,e)$ that are true in all states that result from changing the literals in e true in states that satisfy V and p, where V is a set of 2-literal clauses. Because R_O is monotonic and the universe is finite, there is a fixpoint of R_0 that is obtained by a finite number $n \geq 0$ of iterations as $R^n_O(V_0) = R^{n+1}_O(V_0)$. Here $V_0 = \{a \vee b | a \in I, b \in L\}$ consists of the 2-literal clauses true in the initial state I and L is the set of all literals.

The algorithm does not compute all 2-literal invariants. This is because a 2-literal representation $R^i_O(V_0)$ of a subset of reachable states is not accurate ($s \models R^i_O(V_0)$ does not guarantee that s is reachable), and F_o considers applications of o in unreachable states that result in unreachable states where some 2-literal invariants are violated. In many problem domains, for example in the ones mentioned in this paper, this does not happen.

Lemma 4.1 *For any set of operators $O = \{o_1, \ldots, o_n\}$, the functions F_{o_i}, and consequently the function R_O, are monotonic.*

Lemma 4.2 *Let p and e be sets of literals, V be a set of 2-literal clauses, s be a model such that $s \models V \cup p$, and s' a model obtained from s by setting the literals in e true. Then $s' \models U(V,p,e)$.*

Lemma 4.3 *For all states s and sets of 2-literal clauses V such that $s \models V$ and s' is a successor of s (under the application of an operator in O), $s' \models R_O(V)$.*

Proof: Now $s \models p$ and $s' = s\backslash\{\bar{l} | l \in e\} \cup e$ for some operator $\langle p, e\rangle$. Let $a \vee b$ be any member of $R_O(V)$. Because $R_O(V) \subseteq V$ and $s \models V$, $s \models a \vee b$. Assume $\neg a \notin e$ and $\neg b \notin e$. Clearly $s' \models a \vee b$. So assume $\neg a \in e$ or $\neg b \in e$. Because of symmetry it suffices to consider the case $\neg a \in e$. Because $a \vee b \in R_O(V)$, for all $\langle p', e'\rangle \in O$ such that $V \cup p' \not\models \bot$, $b \in U(V, p', e')$. This holds also for $p' = p, e' = e$. By Lemma 4.2 $s' \models U(V,p,e)$, and hence $s' \models b$. Therefore $s' \models R_O(V)$. □

Theorem 4.4 *Let V be a fixpoint of R_O such that $V \subseteq V_0$. For all states s reachable from the initial state I with applications of operators in O, $s \models V$.*

[1]The mutual exclusion relations on literals in Graphplan [Blum and Furst, 1995] are a superset of the 2-literal invariants.

```
                0123456                    012345
        atR2JFK T FFFFF        loadR1R2JFK  FFFFF
        atR1JFK T FFFFF     loadR1R2London FFF FF
         inR1R2 F  F  F      loadR1R2Paris FFFFFF
     atR2London FF T FF        loadR2R1JFK  FFFFF
     atR1London FF T FF     loadR2R1London FFF FF
      atR2Paris FFFFF T      loadR2R1Paris FFFFFF
      atR1Paris FFFFF T      unloadR1R2JFK FFFFFF
         inR2R1 F  F  F   unloadR1R2London FF FFF
         fuelR1 TT   FF    unloadR1R2Paris FFFFF
  connJFKLondon TTTTTTT      unloadR2R1JFK FFFFFF
   connJFKParis FFFFFFF   unloadR2R1London FF FFF
  connLondonJFK FFFFFFF    unloadR2R1Paris FFFFF
connLondonParis TTTTTTT    moveR1JFKLondon F FFFF
   connParisJFK FFFFFFF     moveR1JFKParis FFFFFF
connParisLondon FFFFFFF    moveR1LondonJFK FFFFFF
         fuelR2 TT   FF  moveR1LondonParis FFFF F
                            moveR1ParisJFK FFFFFF
                         moveR1ParisLondon FFFFFF
                           moveR2JFKLondon F FFFF
                            moveR2JFKParis FFFFFF
                           moveR2LondonJFK FFFFFF
                         moveR2LondonParis FFFF F
                            moveR2ParisJFK FFFFFF
                         moveR2ParisLondon FFFFFF
```

Figure 2: An intermediate stage in finding a plan

Proof: Because by Lemma 4.1 R_O is monotonic, there is a fixpoint V of R_O such that $V \subseteq V_0$. Because s is reachable from I, there is a finite sequence $o_1, o_2, \ldots, o_m$ of operators in O and states $s_0, s_1, \ldots, s_m$ such that $I = s_0$, $s = s_m$ and s_i is obtained from s_{i-1} by the application of o_i. Because $I \models V_0$, by repeated applications of Lemma 4.3 $s_i \models R^i_O(V_0)$ for all $i \in \{0, \ldots, m\}$. Because $V \subseteq R^i_O(V_0)$, $s_i \models V$ for all $i \in \{0, ..., m\}$. Hence $s = s_m \models V$. □

The contents of planning graphs in GraphPlan resemble the intermediate stages in the computation of invariants in our algorithm.

5 AN EXAMPLE

Figure 5 illustrates the solution of a problem from the rocket domain of [Blum and Furst, 1995]. Initially two rockets are at JFK. In the goal state the rockets are in Paris. Flights are possible only from JFK to London and from London to Paris. The algorithm detects the inexistence of plans of lengths 1 to 5 without search only by using the easy polynomial-time inferences. For plan length 6, when the algorithm has reached its first branching point, the state and the known operations are as shown in Figure 5. The remaining possibilities represent the two possible plans, load one of the rockets into the other, fly to London (consuming the fuel from the rocket), unload, load the rocket with no fuel into the one with fuel, fly to Paris, and unload again.

Once the planner decides (arbitrarily) to try to load R1 into R2 at JFK at time 0, the rest of the operations are forced and no contradictions are obtained: R2 has to be flown to London, R1 has to be unloaded from R2, R2 has to be loaded into R1, R1 has to be flown to Paris, and finally R2 has to be unloaded from R1. On the same problem Graphplan generates a search tree with several nodes, because it does not infer anything from the goal state.

6 EXPERIMENTAL RESULTS AND RELATED WORK

We have implemented the algorithm in C, and evaluated it on a number of examples. Tables 1 and 2 lists the runtimes (in seconds) of our planner (the R column), Blum and Furst's *Graphplan* [1995], and the satisfiability programs *Walksat* [Kautz and Selman, 1996], *satz* [Li and Anbulagan, 1997], and *relsat* [Bayardo, Jr. and Schrag, 1997] on a number of benchmark problems from [Kautz and Selman, 1996]. All the SAT programs were run with problem encodings devised by Kautz and Selman. We ran the benchmarks using Li and Anbulagan's *satz* as they do not give runtimes for these benchmarks in their paper. The runs on *satz* and on our planner were on a Sun Ultra 2 workstation. Kautz and Selman ran Graphplan and Walksat on SGI Challenger, and Bayardo and Schrag ran their program on SPARC-10. Blank means that we found no runtime in the papers mentioned. Whenever a runtime was reported for several SAT encodings, we give here the lowest time. For our runs we give the total time taken by our planner, and in parentheses the time taken by the last stage, that is, the time it takes to find a shortest plan if it is known what the length of this plan is. The times in parentheses for the satisfiability algorithms are for the last stage as given in the respective papers. The times for identifying lengths of shortest plans are not given in the satisfiability papers presumably because the algorithms by Kautz and Selman – who started the work on planning by satisfiability – are not capable of determining the inexistence of plans of certain length, that is the unsatisfiability of a set of clauses, with certainty.

The runtimes of our program for the blocks world problems are worse than those by the satisfiability algorithms. Unlike in the other benchmarks, the constraints on the candidate plans at the last stages are very loose, and our reliance on inferences based on failed literal detection does not pay off. The solution of these benchmarks heavily relies on good branching heuristics, which in the case of the satisfiability algorithms are more successful. The more compact encoding of these problems by Kautz and Selman is not the cause of this difference. For example *satz* generates a search tree with only 6 choice nodes for bw-large.c. If we hardwire our program to make the same choices, the solution is found immediately without search. The parallel versions of these blocks world problems are more constrained and solutions are found quickly, for example, solving bw-large.d even with standard parallelism without post-serializability [Dimopoulos *et al.*, 1997] takes 100 seconds. The number of

Table 1: Runtimes of some benchmark problems

problem	R	BF95	KS96
rocket.ext.a	3.8 (1.4)	520	(0.1)
rocket.ext.b	2.7 (0.6)	2337	(0.2)
bw-large.b	74.1 (33.0)	27115	(22)
bw-large.c	7144 (2350)	> 10 h	(564)
bw-large.d	> 10 h	> 10 h	(937)
logistics.a	2.2 (1.4)	6743	(2.7)
logistics.b	91.4 (1.9)	2893	(1.6)
logistics.c	871.2 (3.7)	> 10 h	(1.9)

Table 2: Runtimes of some benchmark problems

problem	R	LA97	BS97
rocket.ext.a	3.8 (1.4)	(0.07)	
rocket.ext.b	2.7 (0.6)	(0.06)	
bw-large.b	74.1 (33.0)	(0.3)	
bw-large.c	7144 (2350)	(1.6)	(11.9)
bw-large.d	> 10 h	(218.9)	(813.3)
logistics.a	2.2 (1.4)	(4.2)	(4.1)
logistics.b	91.4 (1.9)	(0.8)	(16.6)
logistics.c	871.2 (3.7)	(324.2)	(90.3)

choice nodes in the search tree is 5.

Table 4 lists runtimes on a number of benchmarks from Dimopoulos et al. [1997]. Dimopoulos et al. encode planning problems as nonmonotonic propositional logic programs, and find plans by using a program (smodels [Niemelä and Simons, 1996]) that computes the stable models of these programs. The runtimes for Graphplan are by Dimopoulos et al. and all runs by them were on Sparc Ultra. The times in the DNK97 column are for finding a shortest plan when the length of that plan is given as input. This corresponds to the time in our column in parentheses. Like above, our time outside the parentheses is the total runtime. The runtimes in Tables 1 and 2 are not directly comparable to the those in Table 4. The blocks world runtimes are for a parallel encoding, and further, in all the benchmarks, it is not required that parallel operations can be executed in all possible orders, but that there is at least one order in which they can be executed [Dimopoulos *et al.*, 1997]. For example loading or unloading a truck can be performed in parallel with moving it. With this relaxation shortest plans are shorter, the constraints on possible plans are tighter, and search space gets pruned dramatically. Graphplan runtimes are with standard parallelism.

Table 3: Sizes of the search trees

problem	choice nodes			plan	
	total	last	failed	length	ops
rocketext.a	31	7	3	7	34
rocketext.b	26	4	0	7	30
bw-large.b	2	2	0	18	18
bw-large.c	1573	789	785	28	28
bw-large.d					
logistics.a	14	13	0	11	54
logistics.b	438	25	1	13	47
logistics.c	3119	22	1	13	65

Table 4: Runtimes of some benchmark problems

problem	R	BF95	DNK97
bw-large.c	5.9 (5.5)	1830	(190)
bw-large.d	20.4 (19.7)	219600	(157)
bw-large.e	30.4 (29.3)		(365)
logistics.a	0.2 (0.1)	6743	
logistics.b	1.2 (0.2)	2893	
logistics.c	2.7 (0.3)	> 10 h	(18)
train.a	22.7 (21.3)		(647)
train.b	100.1 (96.2)		(1261)
train.c	685.0 (109.2)		(5989)

Table 5: Sizes of the search trees

problem	choice nodes			plan	
	total	last	failed	length	ops
bw-large.c	8	8	0	6	21
bw-large.d	10	10	0	6	32
bw-large.e	10	10	0	7	37
logistics.a	13	13	0	7	68
logistics.b	30	21	0	8	56
logistics.c	39	18	0	8	68
train.a	54	54	45	8	39
train.b	53	53	43	7	34
train.c	410	65	52	8	42

Our runtimes do not include the expansion of operator schemata to sets of propositional operators nor the computation of invariants. These computations take between a second and a minute (bw-large.e) for the problems mentioned. The program performing this is written in Standard ML. An efficient implementation in C would run much faster. Our runtimes include, unlike the runtimes for satisfiability algorithms, a preprocessing stage a major part of which includes the precomputation of pairs of operators that interfere and therefore cannot be applied in parallel.

Tables 3 and 5 list statistics on the sizes of search trees and solutions with our algorithm. The first column is the total number of choice nodes in the search trees, that is, nodes that correspond to case analyses on operations or propositions. The second column is the number of choice nodes at the last stage, and the third column is the number of failed choices at the last stage (at earlier stages all choices fail.) The third column being zero (or small in comparison to the second column) indicates that the heuristics based on failed literal detection was successful in choosing choice points for case analysis. The fourth column gives the length of the plan found, and the fifth the number of operations in the plan. Plan lengths for the other planning and satisfiability algorithms are the same except with the sequential version of blocks worlds problems, where our plans are twice the length of plans obtained with the satisfiability algorithms. This difference is due to the encoding by Kautz and Selman's, that perform a pickup and a putdown operation at the same point of time. The numbers of operations for different algorithms in many cases differ, as the solutions usually are not unique.

An interesting observation is that the solution of many of the problems requires hardly any search. For example on the parallel version of bw-large.e the algorithm detects the inexistence of plans of length 6 and below simply on the basis of what can be inferred from the initial and goal states[2]. All this computation is polynomial time on the size of the planning problem. On plans of length 7 the search tree contains 10 choice nodes, all of which have only one child, that is, the choice was successful. On train.b, on the other hand, finding the plan involves search through several dozens of nodes, but all the failed branches are of height 1, that is, a wrong operator is chosen for application but the error is soon detected by failed literal detection without generating more nodes to the search tree. Search trees of this structure are of polynomial size, and therefore represent polynomial time computation.

Graphplan [Blum and Furst, 1995] is the state-of-the-art STRIPS planner. The main differences between Graphplan and the rest of the work discussed here are that Graphplan is based on backward search from the goal state, and that the mutual exclusion relations that serve the same purpose as invariants are only used for abandoning subgoals not reachable from the initial state, and not for making inferences about the state of affairs at a particular moment of time. These differences seem to be the decisive factor that on many benchmarks makes Graphplan fare worse than the other programs. Note that the information in Graphplan's planning graph until "leveling off" [Blum and Furst, 1995] is stronger than the 2-literal invariants. However, most of this information is inferred by unit simplifications in SAT algorithms and the inference rules in our algorithm. Graphplan employs memoization to avoid repeatedly making the same mistakes, but this does not give an advantage over the rest of the approaches that do not have memoization (excluding *relsat*) on the kind of problems considered here.

[2] Also Kautz and Selman's satplan [1996] sometimes finds out during problem instance generation that there is no solution of a certain length, simply by unit resolution and unit subsumption.

7 CONCLUSIONS

An important research topic is the identification of better heuristics in choosing branching operators. Our current implementation simply chooses the operator (in)application that maximizes the number of propositions or operators that are assigned a truth-value. This is roughly what *satz* does, with the exception that we have nothing that corresponds to counting the number of short clauses, that is, no estimation on the possibilities to use the polynomial time inferences at the next steps is done. In many cases when the constraints imposed by the initial and goal states are not very tight our heuristic chooses an operation that is able to force the longest sequence of (nonsensical) operations which often does not contribute to bridging the gap between the initial state and the goal. Interestingly, when using the algorithm to find a long solution to a problem that has a short solution, this behavior often very quickly and successfully produces the desired result.

In addition to improved branching heuristics, there are prospects of speeding up the algorithm by a more aggressive use of invariants. In addition to 2-literal invariants, some problem domains would benefit from n-literal invariants for $n \geq 3$. Our invariant algorithm can easily be generalized to the $n \geq 3$ case, but it is not clear how it can be made efficient. Also, our algorithm does not take advantage of the fact that usually the set of ground operators is highly structured, and it is often possible to compute the invariants more efficiently directly from a schematic representation of the operators. The representation of a set of candidate invariants in the description of our invariant algorithm consumes a lot of memory (our implementation uses a slightly better representation), and therefore a more concise representation should be used instead, for example one based on binary decision diagrams [Bryant, 1992].

Acknowledgements

We thank Chu Min Li for kindly providing us copies of satz and related satisfiability programs, and Harald Rueß for comments and advice. Financial support from Suomalainen tiedeakatemia and Suomen Kulttuurirahasto is gratefully acknowledged.

References

Bayardo, Jr., R. J. and R. C. Schrag: Using CSP look-back techniques to solve real-world SAT instances. In *Proceedings of the 14th National Conference on Artificial Intelligence*, pages 203–208, 1997.

Bensalem, S., Y. Lakhnech, and H. Saidi: Powerful techniques for the automatic generation of invariants. In Rajeev Alur and Thomas A. Henzinger, editors, *Proceedings of the Eighth International Conference on Computer Aided Verification CAV*, volume 1102 of *Lecture Notes in Computer Science*, pages 323–335, New Brunswick, NJ, USA, July 1996. Springer Verlag.

Blum, A. L. and M. L. Furst: Fast planning through planning graph analysis. In C. S. Mellish, editor, *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1636–1642. Morgan Kaufmann Publishers, 1995.

Bryant, R. E.: Symbolic Boolean manipulation with ordered binary decision diagrams. *ACM Computing Surveys*, 24(3):293–318, September 1992.

Crawford, J. M. and L. D. Auton: Experimental results on the crossover point in random 3-SAT. *Artificial Intelligence*, 81:31–57, 1996.

Davis, M., G. Logemann, and D. Loveland: A machine program for theorem proving. *Communications of the ACM*, 5:394–397, 1962.

Dimopoulos, Y., B. Nebel, and J. Köhler: Encoding planning problems in nonmonotonic logic programs. In *Proceedings of the European Conference on Planning (ECP-97)*. Springer-Verlag, 1997.

Ernst, M., T. Millstein, and D. S. Weld: Tradeoffs in automatic SAT-compilation of planning problems. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, 1997.

Freeman, J. W.: *Improvements to Propositional Satisfiability Search Algorithms*. PhD thesis, University of Pennsylvania, 1995.

Haas, A. R.: The case for domain-specific frame axioms. In F. M. Brown, editor, *The Frame Problem in Artificial Intelligence: Proceedings of the 1987 Workshop*, pages 343–348. Morgan Kaufmann Publishers, 1987.

Kautz, H. and B. Selman: Planning as satisfiability. In B. Neumann, editor, *Proceedings of the 10th European Conference on Artificial Intelligence*, pages 359–363, Wien, Austria, 1992. John Wiley & Sons.

Kautz, H. and B. Selman: Pushing the envelope: planning, propositional logic, and stochastic search. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eight Innovative Applications of Artificial Intelligence Conference*, pages 1194–1201, Menlo Park, California, August 1996. AAAI Press / The MIT Press.

Kautz, H., D. McAllester, and B. Selman: Encoding plans in propositional logic. In L. C. Aiello, J. Doyle, and S. Shapiro, editors, *Proceedings of the 5th International Conference on Principles of Knowledge Representation and Reasoning*, pages 374–385. Morgan Kaufmann Publishers, 1996.

Li, C. M. and Anbulagan: Heuristics based on unit propagation for satisfiability problems. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, August 1997.

McAllester, D. A. and D. Rosenblitt: Systematic nonlinear planning. In T. L. Dean and K. McKeown, editors, *Proceedings of the 9th National Conference on Artificial Intelligence*, pages 634–639, Anaheim, California, 1991. The MIT Press.

Niemelä, I. and P. Simons: Efficient implementation of the well-founded and stable model semantics. In M. Maher, editor, *Proceedings of the 1996 Joint International Conference and Symposium on Logic Programming*, pages 289–303, Bonn, Germany, September 1996. The MIT Press.

Schubert, L.: Monotonic solution of the frame problem in the situation calculus: An efficient method for worlds with fully specified actions. In H. E. Kyburg, R. P. Loui, and G. N. Carlson, editors, *Knowledge Representation and Defeasible Reasoning*, pages 23–68. Kluwer, Boston, 1990.

Efficient Modal Reasoning

More evaluation of decision procedures for modal logics

Enrico Giunchiglia
DIST, V.le Causa 13
16145 Genova, Italy

Fausto Giunchiglia
DISA, Università di Trento.
IRST, 38050 Povo,
Trento, Italy

Roberto Sebastiani
IRST, 38050 Povo,
Trento, Italy

Armando Tacchella
DIST, V.le Causa 13
16145 Genova, Italy

Abstract

This paper follows on previous papers which presented and evaluated various decision procedures for modal logics. It confirms previous experimental results in showing that SAT based decision procedures, i.e., the procedures built on top of decision procedures for propositional satisfiability, are more efficient than tableau based decision procedures. It also confirms previous evidence of an easy-hard-easy pattern in the satisfiability curve for modal K. Finally, it provides further experimental results, suggesting that SAT based decision procedures are also more efficient than the decision procedures based on Ohlbach's translation method. Our results contradict some of the results presented in previous papers.

1 INTRODUCTION

Until recently, the decision procedures for modal logics which could be found in the literature were either tableau based [1] (see, e.g., [Kripke, 1959; Fitting, 1983; Massacci, 1994]), or based on Ohlbach's translation method [Ohlbach, 1991]. Only few of these procedures were implemented, and even less were thoroughly tested and comparatively evaluated. Giunchiglia and Sebastiani [Giunchiglia and Sebastiani, 1996a] changed this situation in three ways. First, they proposed a novel approach where a decision procedure for modal logics, called KSAT, is defined in terms of a SAT decision procedure. To emphasize this fact, they called the resulting class of procedures, SAT based. Second, they provided a new test methodology which naturally extends the fixed-clause-length test method, a very popular test method for SAT [Mitchell *et al.*, 1992; Buro and Buning, 1992]. Third, they built a straightforward LISP implementation of KSAT, called KSATLISP from now on, and tested it against the tableau system KRIS [Hollunder *et al.*, 1990; Baader *et al.*, 1994]. These latter results are reported in [Giunchiglia and Sebastiani, 1996c].

Following on this work, Horrocks presented FACT, a system which further enhances KSAT's ideas by introducing several other optimizations (e.g. backjumping). According to [Horrocks, 1997], the LISP implementation of FACT further enhances KSATLISP performances. Hustadt and Schmidt, instead, make in [Hustadt and Schmidt, 1997a; Hustadt and Schmidt, 1997b] a very detailed and critical analysis of the work in [Giunchiglia and Sebastiani, 1996a; Giunchiglia and Sebastiani, 1996c]. They start by improving on Giunchiglia and Sebastiani's test methodology, thus avoiding possible situations where this method can generate trivial problems. Then, they challenge the following three claims made by Giunchiglia and Sebastiani in [Giunchiglia and Sebastiani, 1996c] (as Hustadt and Schmidt quote them in [Hustadt and Schmidt, 1997a]):

1. KSAT outperforms by orders of magnitude the previous state-of-the-art decision procedures.
2. All SAT based modal decision procedures are intrinsically bound to be more efficient than tableau based decision procedures.
3. There is partial evidence of an easy-hard-easy pattern on randomly generated modal logic formulas independent of all the parameters of evaluation considered.

Finally, they propose TA, a system based on Ohlbach's translation method, which is implemented by appropriately calling FLOTTER and the first order theorem prover SPASS [Weidenbach *et al.*, 1996]. Their analysis suggests that TA has better computational behavior than KSATLISP.

[1] We call "tableau based" any system that implements Smullyan's tableau calculus as defined in [Smullyan, 1968]. Thus, in our terminology, KRIS [Hollunder *et al.*, 1990; Baader *et al.*, 1994] and LWB [Heuerding *et al.*, 1996] are tableau based systems, whence KSAT [Giunchiglia and Sebastiani, 1996a], TA [Hustadt and Schmidt, 1997a] and also FACT [Horrocks, 1997] are not.

The goal of this paper is to provide further experimental results and use them to shed some light on the previous analysis, which, as hinted above, is contradictory in some parts. Thus, in Section 2 we describe KSATC, a C++ implementation of KSAT built starting from a pre-existing state-of-the-art SAT decision procedure. The goal of this section is twofold. First, it shows how the methodology and ideas introduced in [Giunchiglia and Sebastiani, 1996a; Giunchiglia and Sebastiani, 1996c] allows us, in practice, to develop decision procedures for modal logics starting from existing implementations of SAT decision procedures. Given the fact that this latter domain has received much more attention and it is far more developed, we consider this an important result. To build an efficient SAT procedure from scratch requires a lot of man power. Second, a comparison of KSATLISP and KSATC allows us to get an idea of how much we can gain by working on the implementation. [2] Then, in Section 3 we describe a first set of tests which allows us to confirm the three claims of ours listed above and challenged by Hustadt and Schmidt. These tests also suggest that KSATC and (contrarily to what reported in [Hustadt and Schmidt, 1997a; Hustadt and Schmidt, 1997b]) even KSATLISP are faster than TA. Finally, in Section 4 we focus on KSATC and TA. These tests confirm and make more evident the superior computational properties of KSATC over TA.

2 KsatC

We use the same conventions as [Giunchiglia and Sebastiani, 1996c; Giunchiglia and Sebastiani, 1996b]. A *modal atom* is a wff of the form $\Box\psi$, and an *atom* is either a modal atom or a propositional letter. An *assignment* is a partial function from the set of atoms to {*True*, *False*}. In this paper, all the assignments have a finite domain. This allows us to identify an assignment μ as the conjunction of the atoms true in μ and of the negations of the atoms false in μ. The idea is to see a modal wff as a propositional wff in its top-level atoms, so that μ corresponds to a standard assignment in propositional logic.

KSAT is a correct and complete decision procedure for satisfiability in the modal logic K (K-satisfiability from now on) presented and analyzed in detail in [Giunchiglia and Sebastiani, 1996a; Giunchiglia and Sebastiani, 1996c]. A high-level description of KSAT is reported in Figure 2. For any modal wff φ, KSAT(φ) returns *True* if φ is K-satisfiable and *False* otherwise. This is accomplished by calling KSAT$_W(\varphi, \emptyset)$.

[2]To this extent, it must be pointed out that the implementation of KSATC is very similar in spirit to that of KSATLISP. (See Section 2 for the details.) We believe that the behavior of KSATC could be further improved by adding new heuristics, e.g., those implemented in FACT [Horrocks, 1997].

KSAT$_W$ is a variant of the Davis-Putnam-Longemann-Loveland SAT procedure [Davis and Putnam, 1960; Davis *et al.*, 1962] (DPLL from now on). The fundamental difference with DPLL is that, whenever an assignment μ satisfying φ has been found ("base" case), KSAT$_W$ invokes KSAT$_A(\mu)$ instead of returning *True*. Basically, KSAT$_W$ is used to generate a complete set of assignments for φ.[3] Then, the K-consistency of each assignment generated by KSAT$_W$ is checked by KSAT$_A$, and this is done by recursively calling KSAT. These procedures recurse until we get to an assignment with no modal atoms.

KSATC is a C++ implementation of KSAT. KSATC is implemented on top of Böhm's C implementation of DPLL (called DPLL from now on), the winner of a 1992 SAT competition [Buro and Buning, 1992].[4] DPLL has served as the basis for the implementation of KSAT$_W$. The basic features of KSAT$_W$ inherited from DPLL are:

- efficient data structures for literal assignment and wff (partial) evaluation. In DPLL (and hence in KSAT$_W$) literals are assigned and unassigned dynamically inside a wff simply by moving pointers (that is, without copying the wff in the stack at each recursive call (!)) in time proportional to the number of their occurrences.
- smart splitting heuristics. Similarly to other state-of-the-art satisfiability checkers (see, e.g., [Crawford and Auton, 1993]) DPLL splits on the literal occurring most often in the shortest clauses.[5]

KSAT$_W$ has been obtained from DPLL by performing the following steps:

- The code implementing the pure literal rule has been ruled out.[6] With the pure literal rule, the set of assignments generated by KSAT$_W$ might be incomplete, so that KSATC might conclude that the input wff is not satisfiable even if this is actually the case;
- Whenever an assignment μ satisfying φ is found, KSAT$_W$ invokes KSAT$_A(\mu)$ (instead of printing out the assignment);

[3]We say that a set S of assignments is *complete* for a wff ϕ, if ϕ is logically equivalent to the disjunction of the assignments in S.

[4]The code of DPLL and a report about this competition are available at the http address `http://www.informatik.uni-koeln.de/ls_juenger/staff/boehm.html`.

[5]By contrast, in KSATLISP a copy of the wff is passed at each DPLL recursive call; literals are assigned by recursively descending the whole wff; splitting happens on atoms (and not on literals), and KSATLISP chooses the atom occurring most often.

[6]According to the pure literal rule, if a literal l occurs in φ while $\neg l$ does not, l is assigned to *True* [Davis and Putnam, 1960].

```
function KSAT($\varphi$)
    return KSAT$_W$($\varphi$, $T$);


function KSAT$_W$($\varphi, \mu$)
    if $\varphi = T$                                        /* base            */
        then return KSAT$_A$($\mu$);
    if $\varphi = F$                                        /* backtrack       */
        then return False;
    if {a unit clause ($l$) occurs in $\varphi$}             /* unit            */
        then return KSAT$_W$($assign(l, \varphi), \mu \wedge l$);
    if not KSAT$_A$($\mu$)                                   /* early pruning (optional) */
        then return False;
    $l$ := choose-literal($\varphi$);                        /* split           */
    return KSAT$_W$($assign(l, \varphi), \mu \wedge l$) or
           KSAT$_W$($assign(\neg l, \varphi), \mu \wedge \neg l$);


function KSAT$_A$($\bigwedge_i \Box\alpha_i \wedge \bigwedge_j \neg\Box\beta_j \wedge \gamma$)
    for any conjunct "$\neg\Box\beta_j$" do
        if not KSAT($\bigwedge_i \alpha_i \wedge \neg\beta_j$)
            then return False;
    return True;
```

Figure 1: The structure of KSAT.

- Many of the global variables have been made local to the procedures where they are used: this is necessary to avoid interferences between two calls KSAT$_W(\varphi, \emptyset)$ and KSAT$_W(\varphi', \emptyset)$. Analogously for the static variables.

- The (optional) step of early pruning has been included in DPLL (see Figure 2). This step is not mandatory. However, as discussed in [Giunchiglia and Sebastiani, 1996a], early pruning may detect early inconsistencies in the partial assignment so far built and therefore greatly prune the search space.

KSAT$_W$ is interfaced with the rest of the procedure by means of a global look-up table (LUT) which associates a propositional literal B_i (resp. $\neg B_i$) to each distinct modal literal $\Box\varphi_i$ (resp. $\neg\Box\varphi_i$) occurring in φ. Each row in LUT is a pair $\langle B_i, \varphi_i^* \rangle$ (resp. $\langle \neg B_i, \neg\varphi_i^* \rangle$), where φ_i^* is the propositional wff obtained from φ_i by recursively substituting each atom $\Box\varphi_j$ with the corresponding B_j. When invoked on the input wff φ, KSATC initializes LUT and passes φ^* to KSAT. Each time it is invoked on a propositional assignment μ (which represents $\bigwedge_i \Box\alpha_i \wedge \bigwedge_j \neg\Box\beta_j \wedge \gamma$), KSAT$_A$ retrieves from LUT the CNF wffs corresponding to the literals in μ, it merges them and invokes KSAT on the resulting CNF wffs $\wedge_i\alpha_i \wedge \neg\beta_j$'s. Notice that each retrieval requires only constant time.

3 MORE TESTING OF THREE EARLIER CLAIMS

Our goal in this section is to confirm the three claims listed in the introduction, first made in [Giunchiglia and Sebastiani, 1996c] and then challenged by Hustadt and Schmidt.

3.1 THE EVALUATION STRATEGY

We compare five systems, namely: KRIS, KSATLISP, KSATC, KSATLISP(UNSORTED), and TA.[7] KRIS is a tableau based system implemented in LISP [Hollunder *et al.*, 1990; Baader *et al.*, 1994]. Both KSATC and KSATLISPpre-sort input wffs before the main routine is invoked for the first time. This allows for renaming the permutations of a modal atom (e.g., $\Box(A_1 \vee A_2)$ and

[7]TA is available at `http://www.mpi-sb.mpg.de/` in `~hustadt/mdp/`. In particular, we used TA version 1.2, SPASS/FLOTTER version 0.55. KRIS, KSATLISP, KSATC, the random wff generator and detailed instructions about how to reproduce all the tests presented are available at `ftp://ftp.mrg.dist.unige.it/` in `pub/mrg-systems`. (As the following makes clear, KSATLISP(UNSORTED) is KSATLISP with a heuristic turned off).

$\Box(A_2 \vee A_1)$) with the same propositional letter. This step, if not performed, may dramatically affect the performance of the algorithm [Giunchiglia and Sebastiani, 1996a]. KSATLISP(UNSORTED) is KSATLISP with sorting disabled. We test KSATLISP(UNSORTED)(which, we know a priori, is much slower than KSATLISP) in order to reproduce the tests in [Hustadt and Schmidt, 1997a; Hustadt and Schmidt, 1997b]. In fact, as they explicitly state in [Hustadt and Schmidt, 1997a][page 21], Hustadt and Schmidt test KSATLISP with sorting disabled (that is, what in this paper we call KSATLISP-(UNSORTED)).

To test these systems, we use the test methodology described in [Giunchiglia and Sebastiani, 1996a; Giunchiglia and Sebastiani, 1996c] with the improvements suggested in [Hustadt and Schmidt, 1997a].[8] These improvements solve the problem — highlighted in [Hustadt and Schmidt, 1997a] — that, for some choices of the parameters' values, the generated wffs can be trivial or can be significantly simplified by a simple preprocessing step (like that, for instance, which can be optionally activated in both KSATC and KSATLISP). As a matter of fact, the samples that we used throughout our experimental analisys are highly insensitive to preprocessing (i.e. simplifying). We verified this property by running the built-in simplifier of KSATC on every sample of the PS1 test, and then comparing the original sample vs. the simplified one. The result is that there is little or no difference between them.

We can briefly overview our testing methodology as follows. We consider problem sets, i.e., sets of randomly generated $3CNF_K$ wffs. A $3CNF_K$ wff is a conjunction of $3CNF_K$ clauses, where each clause is a disjunctions of three literals. Modal atoms are restricted to have the form $\Box C$, where C is a $3CNF_K$ clause. A $3CNF_K$ wff is randomly generated according to the following parameters: (i) the modal depth d; (ii) the number of clauses L; (iii) the number of propositional variables N; (iv) the probability p with which an atom occurring in a clause of depth > 0 is purely propositional. Following [Hustadt and Schmidt, 1997a], we fix $p = 0$ and modify the generator to avoid multiple occurrences of the same propositional atom inside one clause.

A problem set is thus characterized by a fixed N and d: we let L vary in such a way to empirically cover the "100% satisfiable – 100% unsatisfiable" transition. Then, for each tuple of the four values in a problem set, we randomly generate 100 $3CNF_K$ wffs. These wffs are given in input to the procedure under test. Satisfiability percentages and median CPU times are plotted against the number of clauses L. For practical reasons, a timeout mechanism stops the execution on one problem after 1000 seconds of CPU. A 1000 seconds median time thus means that more than 50% of the total number of samples has exceeded the timeout.

[8]The features and advantages of this methodology are described in [Giunchiglia and Sebastiani, 1996a].

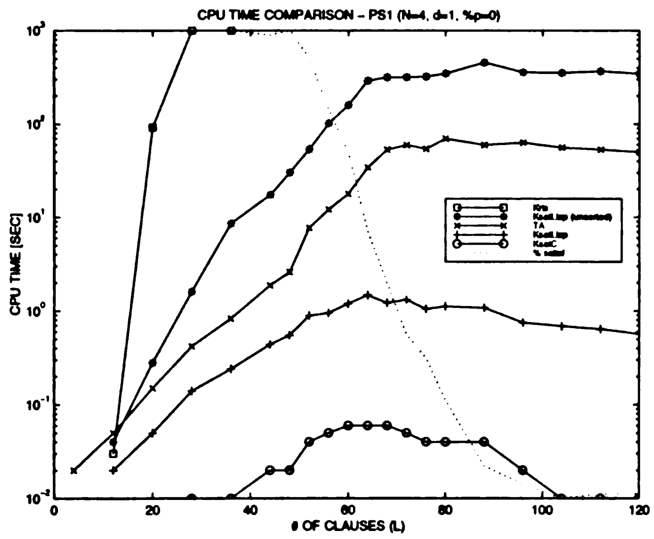
3.2 RESULTS

We start with the problem set PS1 (called PS12 in [Hustadt and Schmidt, 1997a]) characterized by $d = 1$ and $N = 4$. Figure 2 reports the median CPU time for the five systems (left) and the median number of DPLL calls (right) for KSATLISP and KSATC. For KSATLISP and KSATC, the CPU time does not include the time needed for wff sorting. However, this overhead is negligible when compared to the time spent in the decision process. For TA, we consider only the time that SPASS takes to solve the problem. We do not take into account the time to convert modal formulas into first order formulas and the time used by FLOTTER to perform the conversion to conjunctive normal form. We also plot the percentage of satisfiable wffs to get a rough estimation of where the cross-over point of 50% of satisfiable wffs occurs.

Consider first Figure 2 left.[9] Notice the logarithmic scale on the vertical axis. As a first remark, observe the dramatic effect that disabling the sorting of the input formula has on the computational behavior of SAT-based procedures: the gap between KSATLISP-(UNSORTED) and KSATLISP grows up to more than 2 orders of magnitude (e.g., for $L = 80$). The performance gap between KSATLISP and TA is more than 1 order of magnitude for $L \geq 60$. The orders of magnitude become 2 if we consider KSATC instead of KSATLISP. Running TA on highly constrained formulas ($L = 140, 180, 200$), we observed that the median CPU time on 100 samples is always higher than 40 seconds, so we have still no evidence that TA runtime decreases when formulas become trivially unsatisfiable. If we consider KRIS, the gap with KSATLISP and KSATC reaches about 4 and 5 orders of magnitude respectively for $L = 28$. For $L = 36$ no wff is solved by KRIS within the timeout. [10] These results support Claim 1 that KSAT outperforms the other current state-of-the-art decision procedures. With respect to [Giunchiglia and Sebastiani, 1996b] we have further evidence coming from the comparison with TA. These results also show that the arguments against our empirical analysis in [Giunchiglia and Sebastiani, 1996c]

[9]The tests in Figure 2 have been performed on a `Pentium 200MHz MMX, 64MBRam` workstation on `Linux Red-Hat 2.0.30`. SPASS and KSATC are compiled by `gcc 2.7.2.1`, option `-O2`. KRIS and KSATLISP/KSATLISP-(UNSORTED) are compiled by `allegro cl 4.3` and `gcl 2.2` respectively.

[10]For each value of L the test is stopped if more that 50% samples exceed the timeout (i.e., the median value is greater than 1000 seconds). This saves a lot of testing time. Therefore, "no wff is solved within the timeout" means that the first 50%+1 samples all exceeded the timeout, while the others have not been tested.



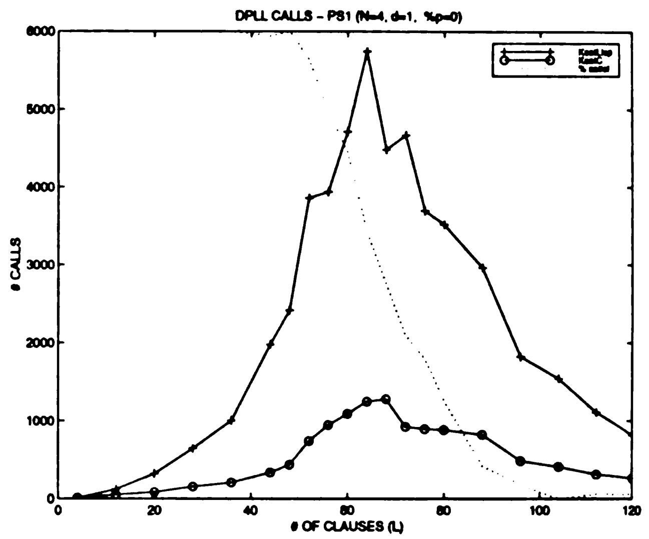


Figure 2: Left: KRIS, TA, KSATLISP(UNSORTED) KSATLISP and KSATC median CPU time, 100 samples/point. Right: KSATLISP and KSATC median search space size, 100 samples/point. Background: satisfiability percentage.

given in [Hustadt and Schmidt, 1997a; Hustadt and Schmidt, 1997b] are wrong. Even fixing the testing methodology, there is still evidence supporting Claim 1. The different results obtained in [Hustadt and Schmidt, 1997a; Hustadt and Schmidt, 1997b] are simply due to the fact that Hustadt and Schmidt tested KSATLISP(UNSORTED) and not KSATLISP, clearly underestimating the effects of this choice.

Consider now Claim 2 that SAT based decision procedures are intrinsically bound to be more efficient than tableau based decision procedures. In [Giunchiglia and Sebastiani, 1996c], (and, more in detail, in [Giunchiglia and Sebastiani, 1996b]) this strong claim was supported by some empirical results and also by a theoretical analysis. We have discussed the empirical results in the above paragraph. For what concerns the theoretical analysis given in [Giunchiglia and Sebastiani, 1996c; Giunchiglia and Sebastiani, 1996b], we still support it. The argument goes as follows. We start from the observation that SAT deciders are intrinsically superior to propositional tableaux, for

(*a*) they don't generate redundant assignments, and
(*b*) they prune branches as soon as they violate one propositional constraint.

Then, it is sufficient to notice that, as [Giunchiglia and Sebastiani, 1996c; Giunchiglia and Sebastiani, 1996b] show, this performance gap propagates and expands with the modal depth of the formulas.

Finally, consider Claim 3 about the existence of easy-hard-easy patterns. From a theoretical point of view, it is shown in [Giunchiglia and Sebastiani, 1996b] that these patterns are a consequence of property (b) above.

From an empirical point of view, consider Figure 2, right. This figure reports the size of the search space for both KSATLISP and KSATC. It is easy to notice the existence of easy-hard-easy peaks centered about the 50% satisfiable point. This issue will be further discussed in Section 4.

4 KsatC VS. TA

Consider now the three problem sets PS2 ($d = 1, N = 4$), PS3 ($d = 1, N = 5$) and PS4 ($d = 1, N = 6$). PS2 and PS4 reproduce the two experiments presented in [Hustadt and Schmidt, 1997a], with a slight change in the testing methodology. In particular, the improved test methodology presented in [Hustadt and Schmidt, 1997a] and described in Section 3 is not yet optimal, as it may be the case that multiple occurrences of the same modal atom $\Box C$ (or different permutations of $\Box C$) occur in clauses of depth > 0. For instance, the modal clause $\Box(A_1 \vee A_2 \vee A_3) \vee \neg\Box(A_1 \vee A_3 \vee A_2)$ becomes a tautology after sorting. To avoid this, we have introduced a further slight improvement in the wff generator: pre-sort all modal atoms and avoid multiple occurrences of (sorted) modal atoms inside clauses. It is clear that there is no point in trying to simplify these samples.

The results of PS2, PS3, PS4 are described in Figure 3.[11] In all figures, the horizontal axis represents the number of clauses L, and each point represent the median value out of 100 values. The three columns cor-

[11]All the tests of Figure 3 have been performed with the same machine, operating system and compilers as those in Figure 2.

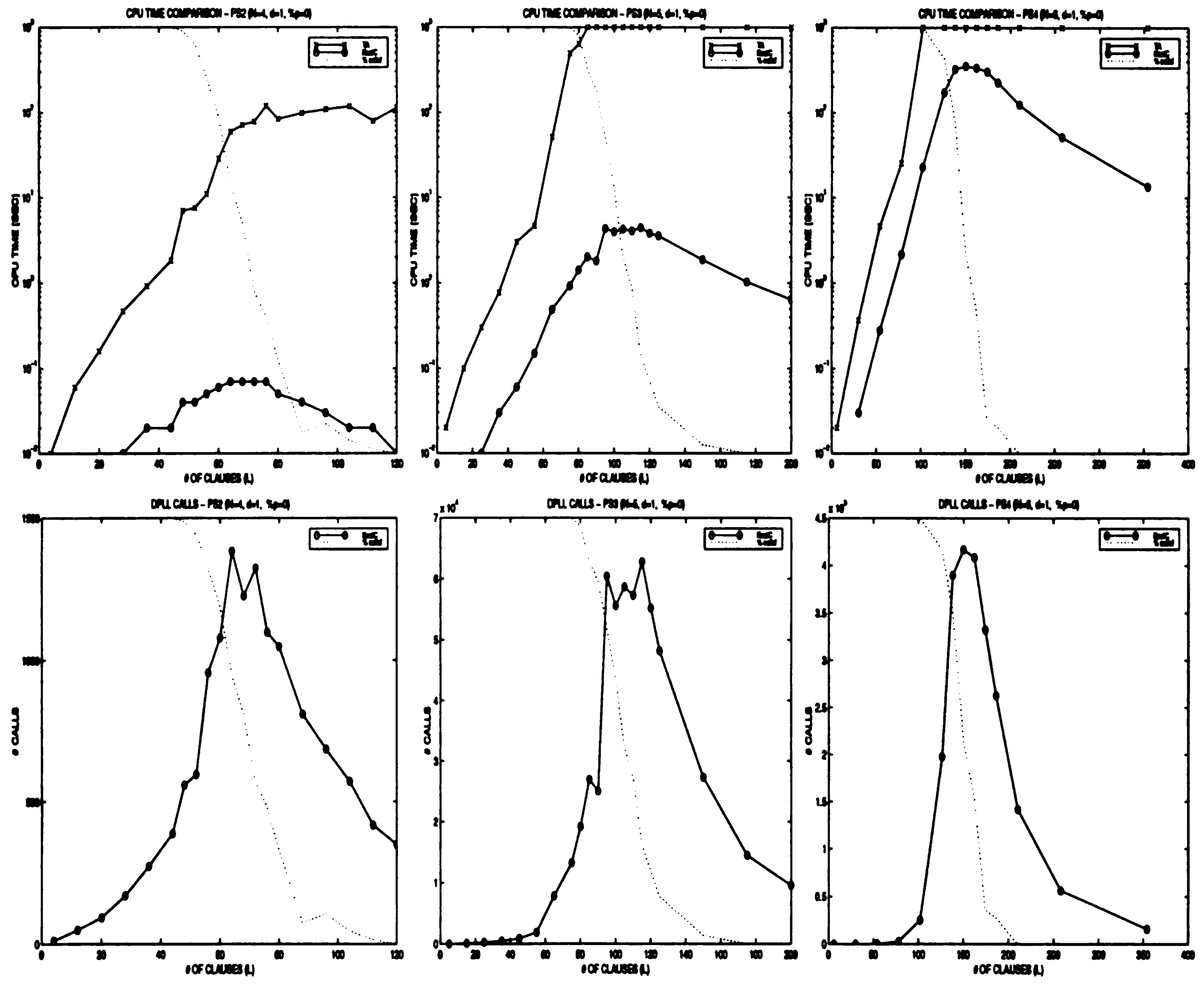


Figure 3: First row: TA versus KSATC on PS2, PS3, PS4. Second row: KSATC size of search space. Background: satisfiability percentages (100% satisfiable at end-scale).

respond to PS2, PS3 and PS4 respectively. In the first row we compare TA and KSATC CPU times, in logarithmic scale. In the second row we plot the KSATC global number of recursive DPLL calls, i.e., the size of the space effectively searched by KSATC, in linear scale. The dotted plots in the background represent the percentages of satisfiable formulas, in linear scale, with 100% satisfiable at end scale.

An eye-catching observation to the first row shows that KSATC performs better than TA in all the three tests. For PS2, the gap between TA and KSATC is of 2 (almost 3) orders of magnitude for $L = 68$ (i.e. at the cross-over point of 50% of satisfiable wffs) and goes up till 4 orders of magnitude at the right end side of the horizontal axis. For PS3 and PS4, TA median values exceed the timeout for $L = 85$ and 102 respectively, while the corresponding values of KSATC are 2.0 and 22.9 seconds. TA keeps exceeding the timeout for all the successive values. After $L = 125$ and 162 respectively, no wff is solved by TA within the timeout.

It is very important to notice the ***qualitative*** difference between the KSATC and TA plots: when the tests enter the "unsatisfiable" area, the KSATC curves ***decrease*** with L. In fact the size of the space searched (bottom row) tends to zero whenever L approaches the "100% unsatisfiable" area. [Giunchiglia and Sebastiani, 1996b] showed that this feature is a consequence of the ability of a procedure to detect constraint violations as soon as they occur: the more constrained the formula is, the more likely a branch violates a constraint, the higher the search tree is pruned. Roughly speaking, this matches the general intuition that overconstrainedness makes unsatisfiability more evident, and thus easier to detect

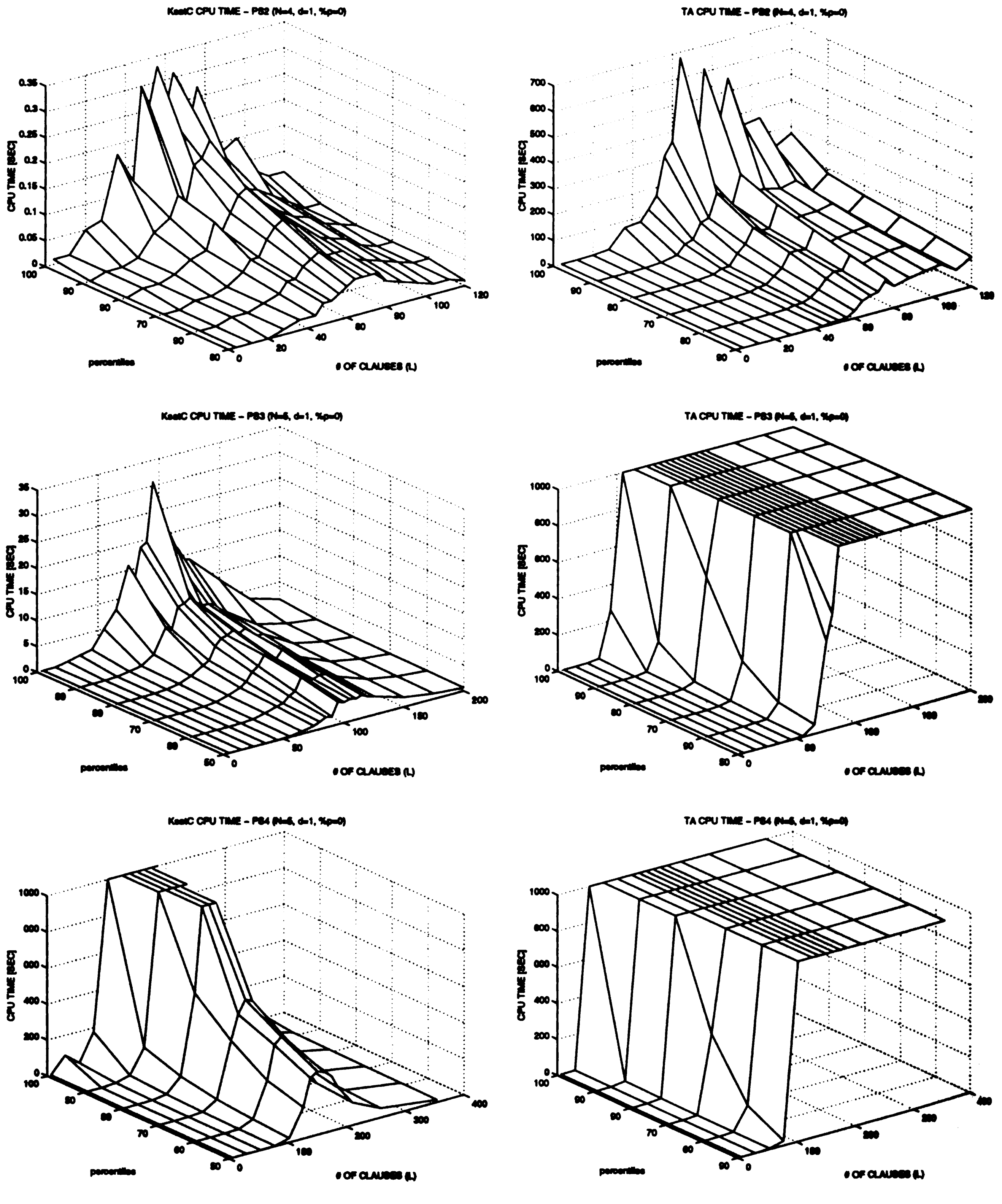


Figure 4: The percentile graphs for KSATC (left column) and TA (right column) on PS2 (top row), PS3 (middle row) and PS4 (bottom row).

(see [Williams and Hogg, 1994] for a fine-grained analysis of this point.) Similarly to tableau based systems [Giunchiglia and Sebastiani, 1996b], TA does not seem capable to take full advantage of over-constrainedness of the input wffs. For instance, in the extreme right points of some plots, TA is not able to solve one single instance within the timeout, while these instances are easy or even trivial to solve for KSATC. To support this consideration we generated a sample $L = 200, N = 5, d = 1$ and tested TA without time limits. TA came out with a response after 9534 seconds (more than two hours and a half) while KSATC took 0.88 seconds of CPU.

It is also worth noticing that the plots support the evidence of a phase transition phenomenon – i.e., steep satisfiability transition plots and easy-hard-easy hardness patterns whose peaks are located about the 50% satisfiability point – for K-satisfiability. This confirms the analogous results in [Giunchiglia and Sebastiani, 1996c; Giunchiglia and Sebastiani, 1996b].

All the above considerations are confirmed by the Q%-percentile graphs in Figure 4. Formally, the *Q%-percentile* of a set S of values is the value V such that Q% of the values in S are smaller or equal to V. The median value of a set thus corresponds to the 50% percentile of the set. Figure 4 reports the 50%, 60%, 70%, 80%, 90% and 100% percentile curves for KSATC (left column) and TA (right column) on PS2 (top row), PS3 (middle row) and PS4 (bottom row). All the Q%-percentile plotted curves show that KSATC performs better than TA (notice the different end of scale used on the vertical axis for KSATC and TA), except for those points in PS4 on which both KSATC and TA exceed the time limit. However, on the basis of the respective values of the surrounding points, we may conjecture that even for these points KSATC values are less than the respective TA values. Interestingly, for PS2, there does not seem to be much difference between the 50%-percentile and the 90%-percentile curves for TA. On PS2, TA 100%-percentile curve seems to present an easy-hard-easy pattern, but this is the only curve where this seems to happen.

To extend our comparison, we have further tested TA and KSATC on the class of wffs $\{\varphi_d^K\}_{d=1,2,\ldots}$ presented in [Halpern and Moses, 1992]. These are K-satisfiable wffs, with depth d and $2d+1$ propositional letters. This test is interesting as every Kripke structure satisfying φ_d^K has at least $2^{d+1}-1$ distinct states, while $|\varphi_d^K|$ is $O(d^2)$. From the results in [Halpern and Moses, 1992] we can reasonably assume a minimum exponential growth factor of 2^d in time for any ordinary algorithm based on Kripke semantics. We have run TA and KSATC on these formulas, for increasing values of d.

Figure 5 presents the resulting CPU times.[12] TA

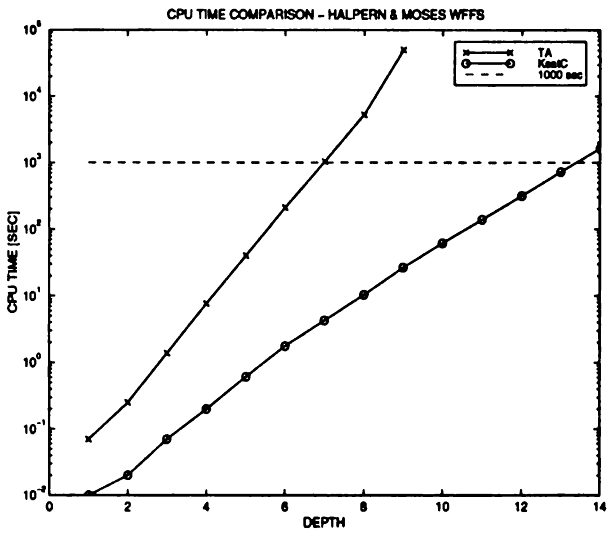


Figure 5: TA and KSATC CPU times for the class of φ_d^K formulas.

and KSATC CPU times grow approximately as $(5.0)^d$ and $(2.3)^d$, respectively, exceeding the 1000 seconds limit for $d = 7$ (1032.35 seconds) and $d = 14$ (1616.87 seconds), respectively. This confirms the performance gap highlighted above. On the same test, on the same machine, KRIS and KSATLISP grow approximatively as $(12.7)^d$ and $(2.6)^d$, respectively. They exceed the 1000 seconds limit for $d = 7$ (5085 seconds) and $d = 11$ (2541.28 seconds) respectively, as reported in [Giunchiglia and Sebastiani, 1996c].

5 CONCLUSION

Our analysis in this paper suggests that SAT based decision procedures have superior behavior than decision procedures based on the translation method. In previous papers we have presented results, confirmed by the results in this paper, which suggest that SAT based decision procedures are also more efficient than tableau based decision procedures. So far we have restricted ourselves to few basic modal logics (e.g., K, S5). An interesting open problem is whether these (substantial) computational advantages will also exist in other modal logics which are more "interesting" from an applicational point of view, e.g., CTL, LTL, Dynamic Logics.

ACKNOWLEDGEMENTS

The work described in this paper has benefitted a lot from the interactions with Ullrich Hustadt and Renate Schmidt. The results published in their papers

[12]The test in Figure 5 have been performed on a SUN SPARC10, 32MBRam workstation on SunOS 4.1.4. SPASS and KSATC are compiled by gcc 2.7.2, option -O2.

[Hustadt and Schmidt, 1997a; Hustadt and Schmidt, 1997b] challenged us and forced us to look deeper into the previous analisys and results. Furthermore, the help of Hustadt has been crucial in the generation of some of the results described in this paper.

This work has also benefitted of many useful discussions with Ian Horrocks, who provided very useful feedback.

The MRG group at Genova put up with many weeks of time-consuming tests.

Finally, thanks to the anonymous referees whose criticisms helped us during the writing of the revised version.

References

[Baader *et al.*, 1994] F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H.J. Profitlich. An Empirical Analysis of Optimization Techniques for Terminological Representation Systems or: Making KRIS get a move on. *Applied Artificial Intelligence. Special Issue on Knowledge Base Management*, 4:109–132, 1994.

[Buro and Buning, 1992] M. Buro and H. Buning. Report on a SAT competition. Technical Report 110, University of Paderborn, Germany, November 1992.

[Crawford and Auton, 1993] J. Crawford and L. Auton. Experimental results on the crossover point in satisfiability problems. In *Proc. of the 11th National Conference on Artificial Intelligence*, pages 21–27, 1993.

[Davis and Putnam, 1960] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7:201–215, 1960.

[Davis *et al.*, 1962] M. Davis, G. Longemann, and D. Loveland. A machine program for theorem proving. *Journal of the ACM*, 5(7), 1962.

[Fitting, 1983] M. Fitting. *Proof Methods for Modal and Intuitionistic Logics.* D. Reidel Publishg, 1983.

[Giunchiglia and Sebastiani, 1996a] F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedures - the case study of modal K. In *Proc. of the 13th Conference on Automated Deduction*, Lecture Notes in Artificial Intelligence, New Brunswick, NJ, USA, August 1996. Springer Verlag. Also DIST-Technical Report 96-0037 and IRST-Technical Report 9601-02.

[Giunchiglia and Sebastiani, 1996b] F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedures - the case study of modal K(m). Technical Report 9611-06, IRST, Trento, Italy, 1996. A shorter version will appear on *Information and Computation.*

[Giunchiglia and Sebastiani, 1996c] F. Giunchiglia and R. Sebastiani. A SAT-based decision procedure for ALC. In *Proc. of the 5th International Conference on Principles of Knowledge Representation and Reasoning - KR'96*, Cambridge, MA, USA, November 1996. Also DIST-Technical Report 9607-08 and IRST-Technical Report 9601-02.

[Halpern and Moses, 1992] J.Y. Halpern and Y. Moses. A guide to the completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54(3):319–379, 1992.

[Heuerding *et al.*, 1996] A. Heuerding, G. Jager, S. Schwendimann, and M. Seyfried. The logics workbench LWB: A snapshot. *Euromath Bulletin*, 2(1):177–186, 1996.

[Hollunder *et al.*, 1990] B. Hollunder, W. Nutt, and M. Schmidt-Schauß. Subsumption Algorithms for Concept Description Languages. In *Proc. 8th European Conference on Artificial Intelligence*, pages 348–353, 1990.

[Horrocks, 1997] Ian R. Horrocks. *Optimising Tableaux Procedures for Description Logics.* PhD thesis, Department of Computer Science, University of Manchester, 1997.

[Hustadt and Schmidt, 1997a] U. Hustadt and R.A. Schmidt. On evaluating decision procedures for modal logic. Research report MPI-I-97-2-003, Max-Planck-Institut für Informatik, Saarbrücken, Germany, February 1997.

[Hustadt and Schmidt, 1997b] U. Hustadt and R.A. Schmidt. On evaluating decision procedures for modal logic. In *Proc. of the 15th International Joint Conference on Artificial Intelligence*, 1997.

[Kripke, 1959] S. Kripke. *A completeness theorem in modal logic*, volume 24. 1959.

[Massacci, 1994] F. Massacci. Strongly analytic tableaux for normal modal logics. In *Proc. of the 12th Conference on Automated Deduction*, 1994.

[Mitchell *et al.*, 1992] D. Mitchell, B. Selman, and H. Levesque. Hard and Easy Distributions of SAT Problems. In *Proc. of the 10th National Conference on Artificial Intelligence*, pages 459–465, 1992.

[Ohlbach, 1991] H. J. Ohlbach. Semantics based translation methods for modal logics. *Journal of Logic and Computation*, 1(5):691–746, 1991.

[Smullyan, 1968] R. M. Smullyan. *First-Order Logic.* Springer-Verlag, NY, 1968.

[Weidenbach *et al.*, 1996] C. Weidenbach, B. Gaede, and G. Rock. SPASS & FLOTTER version 0.42. In M.A. McRobbie and J.K. Slaney, editors, *Proceedings of the 13th Conference on Automated Deduction (CADE-13)*, volume 1104 of *Lecture Notes in Artificial Intelligence*, pages 141–145, New Brunswick, New Jersey, USA, July/August 1996. Springer.

[Williams and Hogg, 1994]
C. P. Williams and T. Hogg. Exploiting the deep structure of constraint problems. *Artificial Intelligence*, 70:73–117, 1994.

Using an Expressive Description Logic: FaCT or Fiction?

Ian R. Horrocks
Medical Informatics Group
Department of Computer Science
University of Manchester
Manchester M13 9PL, UK
horrocks@cs.man.ac.uk www.cs.man.ac.uk/~horrocks

Abstract

Description Logics form a family of formalisms closely related to semantic networks but with the distinguishing characteristic that the semantics of the concept description language is formally defined, so that the subsumption relationship between two concept descriptions can be computed by a suitable algorithm. Description Logics have proved useful in a range of applications but their wider acceptance has been hindered by their limited expressiveness and the intractability of their subsumption algorithms. This paper addresses both these issues by describing a sound and complete tableaux subsumption testing algorithm for a relatively expressive Description Logic which, in spite of the logic's worst case complexity, has been shown to perform well in realistic applications.

1 INTRODUCTION

Description Logics (DLs) form a family of formalisms which have grown out of knowledge representation techniques using frames and semantic networks; their distinguishing characteristic is a formally defined semantics which enables the subsumption (kind-of) relationship to be computed by a suitable algorithm (Woods and Schmolze 1992). DL based knowledge representation systems have proved useful in a range of applications (Berman et al. 1994; Guha and Lenat 1994; Goble et al. 1996; Levy and Rousset 1996; Küssner 1997), but their wider acceptance has been hindered by their limited expressiveness (Doyle and Patil 1991; MacGregor 1991) and the intractability of their subsumption algorithms (Heinsohn et al. 1994; Speel et al. 1995).[1] This paper addresses both these issues by describing a tableaux subsumption testing algorithm for a relatively expressive DL which, in spite of the logic's worst case complexity, has been shown to perform well in realistic applications.

A particularly promising application domain for DLs is in the growth area of ontological engineering—the design, construction and maintenance of large conceptual schemas or ontologies (Mays et al. 1996; Horrocks et al. 1996; Rector and Horrocks 1997). An example of such an application is the European GALEN project, which aims to promote the sharing and re-use of medical data by building a large medical terminology ontology which can be used by application designers as a flexible and extensible classification schema (Rector et al. 1993). However design studies at the outset of the project identified expressive requirements which were satisfied by few if any implemented DL systems (Nowlan 1993), in particular the ability to reason about transitive part-whole, causal and compositional relations (called *roles* in DLs) (Rector et al. 1997). The importance of reasoning with transitive roles has long been recognised (Hors 1994; Padgham and Lambrix 1994; Artale et al. 1996), and has been identified as a requirement in other application domains, particularly those concerned with complex physically composed objects, e.g., engineering (Sattler 1995).

The work presented here was motivated by the desire to provide a sound, complete and empirically tractable algorithm for a DL with the expressive power required by these kinds of application. The logic developed for this purpose was $\mathcal{ALCH}_{R^+}$, an extension of the well known $\mathcal{ALC}$ DL (Schmidt-Schauß and Smolka 1991). $\mathcal{ALC}$ supports concept descriptions using the standard logical connectives, plus existential and universal role restrictions, but only using simple primitive roles. $\mathcal{ALCH}_{R^+}$ augments $\mathcal{ALC}$ with transitively closed primitive roles and role inclusion axioms, the combination of which enables a hierarchy of transitive and non-transitive roles to be defined—a fundamental requirement of the GALEN project—and allows reasoning with respect to general terminologies (see Section 5), a feature which is also required by some applications (Levy

[1] A desire/requirement for sound and complete reasoning is assumed, but see (Borgida 1992) for a discussion of this issue.

and Rousset 1996). A tableaux satisfiability testing algorithm for $\mathcal{ALCH}_{R^+}$ will be presented, along with a proof of its soundness and completeness, and an extension to the algorithm which supports reasoning with attributes (functional/feature roles) will also be described. A highly optimised implementation of this algorithm forms the basis for a terminological classifier, FaCT, which has been developed to demonstrate the feasibility of using the algorithm for subsumption reasoning in realistic applications.

2 TRANSITIVE ROLES

Extensions to $\mathcal{ALC}$ which support some form of transitive roles include $\mathcal{CIQ}$ (Giacomo and Lenzerini 1996), $\mathcal{TSL}$ (Schild 1991), $\mathcal{ALC}_+$ (Baader 1990), $\mathcal{ALC}_{R^+}$ and $\mathcal{ALC}_\oplus$ (Sattler 1996). Of these, $\mathcal{CIQ}$, $\mathcal{TSL}$ and $\mathcal{ALC}_+$ all support role expressions with transitive or transitive reflexive operators, and from correspondence to propositional dynamic logics their satisfiability problems are known to be EXPTIME-complete (Schild 1991). The $\mathcal{ALC}_{R^+}$ and $\mathcal{ALC}_\oplus$ DLs support transitive roles without providing a general transitive closure operator, and were investigated in the hope that a more restricted form of transitive role might lead to a satisfiability problem in a lower complexity class (Sattler 1996).

$\mathcal{ALC}_{R^+}$ augments $\mathcal{ALC}$ with transitively closed primitive roles: an $\mathcal{ALC}_{R^+}$ terminology may include axioms of the form $R \in \mathbf{R}_+$, where R is a role name and $\mathbf{R}_+$ is the set of transitive roles names in the terminology. In (Sattler 1996) an algorithm for deciding the satisfiability of $\mathcal{ALC}_{R^+}$ concept expressions is presented along with a proof of its soundness and completeness. It is also demonstrated that the complexity of the $\mathcal{ALC}_{R^+}$ satisfiability problem is PSPACE-complete, the same as for $\mathcal{ALC}$ (Donini et al. 1995). $\mathcal{ALC}_\oplus$ extends $\mathcal{ALC}_{R^+}$ by associating each non-transitive role R with its transitive orbit. The transitive orbit of a role R, denoted $R^\oplus$, is a transitive role which subsumes R, and could be defined by the axioms $R^\oplus \in \mathbf{R}_+$ and $R \sqsubseteq R^\oplus$. The interpretation of $R^\oplus$ is therefore a superset of the interpretation of the transitive closure of R, i.e., $(R^\oplus)^\mathcal{I} \supseteq (R^+)^\mathcal{I}$. Unfortunately, the complexity of the $\mathcal{ALC}_\oplus$ satisfiability problem is also shown to be EXPTIME-complete.

$\mathcal{ALCH}_{R^+}$ generalises $\mathcal{ALC}_\oplus$ by supporting transitively closed primitive roles and arbitrary role inclusion axioms of the form $R \sqsubseteq S$. As it is more general than $\mathcal{ALC}_\oplus$, but still less expressive than DLs such as $\mathcal{ALC}_+$ which support the transitive closure role forming operator, the $\mathcal{ALCH}_{R^+}$ satisfiability problem is clearly also EXPTIME-complete. However, the tableaux satisfiability testing algorithm for $\mathcal{ALCH}_{R^+}$ is much simpler than for $\mathcal{ALC}_+$:

1. It is simpler to detect cycles in the tableaux construction which could lead to non-termination. Cycle detection (*blocking*) involves comparing sets of concepts, and this is complicated in $\mathcal{ALC}_+$ by the need to deal with concepts containing role expressions. It can be shown, for example, that identifying equivalent role expressions is itself a PSPACE-complete problem (Baader 1990).

2. It is simpler to deal with cycles once they have been detected, because in the $\mathcal{ALCH}_{R^+}$ algorithm all cycles lead to a valid cyclical model. In the $\mathcal{ALC}_+$ algorithm, on the other hand, it is necessary to differentiate between cycles which lead to a valid cyclical model (*good cycles*) and those which do not (*bad cycles*).

$\mathcal{ALCH}_{R^+}$ is sufficiently expressive to be useful in a range of applications, but the simplicity of its satisfiability testing algorithm means that it is easy to implement and amenable to a wide range of optimisation techniques.

3 THE $\mathcal{ALCH}_{R^+}$ DESCRIPTION LOGIC

In this section a tableaux algorithm for testing the satisfiability of $\mathcal{ALCH}_{R^+}$ concept expressions will be described and a proof of its soundness and completeness presented. The algorithm and proof are extensions of those described for $\mathcal{ALC}_{R^+}$ (Sattler 1996).

3.1 SYNTAX AND SEMANTICS

$\mathcal{ALCH}_{R^+}$ is the DL obtained by augmenting $\mathcal{ALC}$ with transitively closed primitive roles and primitive role introduction axioms. An $\mathcal{ALCH}_{R^+}$ terminology consists of a finite set of axioms of the form $C \sqsubseteq D \mid C \doteq D \mid R \sqsubseteq S \mid R \in \mathbf{R}_+$, where C and D are concept expressions, R and S are role names and $\mathbf{R}_+$ is the set of transitive role names. $\mathcal{ALCH}_{R^+}$ concept expressions are of the form $\mathsf{CN} \mid \top \mid \bot \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C$, where CN is a concept name, C and D are concept expressions and R is a role name.

A standard Tarski style model theoretic semantics is used to interpret concept expressions and to justify subsumption inferences (Baader et al. 1991). The meaning of concepts and roles is given by an interpretation $\mathcal{I}$ which is a pair $(\Delta^\mathcal{I}, \cdot^\mathcal{I})$, where $\Delta^\mathcal{I}$ is the domain (a set) and $\cdot^\mathcal{I}$ is an interpretation function. The interpretation function maps each concept name CN to a subset of $\Delta^\mathcal{I}$ and each role to to a set valued function (or equivalently a binary relation): $R^\mathcal{I} : \Delta^\mathcal{I} \longrightarrow 2^{\Delta^\mathcal{I}}$ $(R^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I})$. The semantics of an $\mathcal{ALCH}_{R^+}$ concept expression is derived from the semantics of its components as shown in Figure 1. The semantics of $\mathcal{ALCH}_{R^+}$ axioms is given in Figure 2.

To simplify the description of the algorithm, it will be as-

Syntax	Semantics
CN	$\mathsf{CN}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
$\top$	$\Delta^{\mathcal{I}}$
$\bot$	$\emptyset$
$\neg C$	$\Delta^{\mathcal{I}} - C^{\mathcal{I}}$
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$\exists R.C$	$\{d \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(d) \cap C^{\mathcal{I}} \neq \emptyset\}$
$\forall R.C$	$\{d \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(d) \subseteq C^{\mathcal{I}}\}$

Figure 1: Semantics of $\mathcal{ALCH}_{R^+}$ Concept Expressions

Syntax	Semantics
$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
$C \doteq D$	$C^{\mathcal{I}} = D^{\mathcal{I}}$
$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
$R \in \mathbf{R}_+$	$R^{\mathcal{I}} = (R^{\mathcal{I}})^+$

Figure 2: Semantics of $\mathcal{ALCH}_{R^+}$ Axioms

sumed that $\mathbf{R}_+$ and the $\sqsubseteq$ relation have been defined by an $\mathcal{ALCH}_{R^+}$ terminology $\mathcal{T}$, so that $\mathbf{R}_+ = \{R \mid R \in \mathbf{R}_+$ is an axiom in $\mathcal{T}\}$, and for two roles R and S, $R \sqsubseteq S$ iff $R \sqsubseteq S$ is an axiom in $\mathcal{T}$ or there is a role R' such that $R \sqsubseteq R'$ is an axiom in $\mathcal{T}$ and $R' \sqsubseteq S$. Without loss of generality, it will also be assumed that the concept expression is in negation normal form, so that negations are applied only to concept names, and that the terminology does not contain any concept axioms (i.e., axioms of the form $C \sqsubseteq D$ or $C \doteq D$), so that all concept names are atomic primitives.[2] Arbitrary concept expressions can be transformed into negation normal form using a combination of DeMorgan's laws and the identities $\neg\exists R.C = \forall R.\neg C$ and $\neg\forall R.C = \exists R.\neg C$. How the algorithm can be used to test satisfiability w.r.t. a general terminology will be described in Section 5.

Like other tableaux algorithms, the $\mathcal{ALCH}_{R^+}$ algorithm tries to prove the satisfiability of a concept expression D by demonstrating a model of D—an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ such that $D^{\mathcal{I}} \neq \emptyset$. The model is represented by a tree whose nodes correspond to individuals in the model, each node being labelled with a set of $\mathcal{ALCH}_{R^+}$-concepts. When testing the satisfiability of an $\mathcal{ALCH}_{R^+}$-concept D, these sets are restricted to subsets of $sub(D)$, where $sub(D)$ is the closure of the subexpressions of D defined as follows:

1. if D is an atomic primitive concept or its negation, then $sub(D) = \{D\}$
2. if D is of the form $\exists R.C$ or $\forall R.C$, then $sub(D) = \{D\} \cup sub(C)$
3. if D is of the form $C_1 \sqcap C_2$ or $C_1 \sqcup C_2$, then $sub(D) = \{D\} \cup sub(C_1) \cup sub(C_2)$

[2]An atomic primitive is a concept name CN for which there is no definition in $\mathcal{T}$: all that is known about CN is that $\mathsf{CN}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$.

The soundness and completeness of the algorithm will be proved by showing that a concept has a model iff it has a *tableau*, and that the algorithm constructs a tableau for a concept iff one exists.

Definition 1 If D is an $\mathcal{ALCH}_{R^+}$-concept and $\mathbf{R}_D$ is the set of role names occurring in D, a tableau T for D is defined to be a triple $(\mathbf{S}, \mathcal{L}, \mathcal{E})$ such that: $\mathbf{S}$ is a set of individuals, $\mathcal{L} : \mathbf{S} \to 2^{sub(D)}$ maps each individual to a set of concept expressions which is a subset of $sub(D)$, $\mathcal{E} : \mathbf{R}_D \to 2^{\mathbf{S}\times\mathbf{S}}$ maps each role name occurring in D to a set of pairs of individuals, and there is some individual $s \in \mathbf{S}$ such that $D \in \mathcal{L}(s)$. For all $s \in \mathbf{S}$ it holds that:

1. $\bot \notin \mathcal{L}(s)$, and if $C \in \mathcal{L}(s)$, then $\neg C \notin \mathcal{L}(s)$
2. if $C_1 \sqcap C_2 \in \mathcal{L}(s)$, then $C_1 \in \mathcal{L}(s)$ and $C_2 \in \mathcal{L}(s)$
3. if $C_1 \sqcup C_2 \in \mathcal{L}(s)$, then $C_1 \in \mathcal{L}(s)$ or $C_2 \in \mathcal{L}(s)$
4. if $\forall R.C \in \mathcal{L}(s)$ and $\langle s, t\rangle \in \mathcal{E}(R)$, then $C \in \mathcal{L}(t)$
5. if $\exists R.C \in \mathcal{L}(s)$, then there is some $t \in \mathbf{S}$ s.t. $\langle s, t\rangle \in \mathcal{E}(R)$ and $C \in \mathcal{L}(t)$
6. if $\forall R.C \in \mathcal{L}(s)$, $\langle s, t\rangle \in \mathcal{E}(S)$, $S \in \mathbf{R}_+$ and $S \sqsubseteq R$, then $\forall S.C \in \mathcal{L}(t)$
7. if $R \sqsubseteq S$ then $\mathcal{E}(R) \subseteq \mathcal{E}(S)$

Lemma 1 *An $\mathcal{ALCH}_{R^+}$-concept D is satisfiable iff there exists a tableau for D.*

Proof: For the *if* direction, if $T = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ is a tableau for D, a model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of D can be defined as:

$$\begin{array}{rcl} \Delta^{\mathcal{I}} & = & \mathbf{S} \\ \mathsf{CN}^{\mathcal{I}} & = & \{s \mid \mathsf{CN} \in \mathcal{L}(s)\} \\ & & \text{for all concept names } \mathbf{CN} \text{ in } sub(D) \\ R^{\mathcal{I}} & = & \begin{cases} \mathcal{E}(R)^+ & \text{if } R \in \mathbf{R}_+ \\ \mathcal{E}(R) \cup \bigcup_{S \sqsubseteq R} S^{\mathcal{I}} & \text{otherwise} \end{cases} \end{array}$$

where $\mathcal{E}(R)^+$ denotes the transitive closure of $\mathcal{E}(R)$.

By induction on the structure of concepts it can be shown that $\mathcal{I}$ is well defined and that $D^{\mathcal{I}} \neq \emptyset$. For concepts of the form $\neg C$, $C_1 \sqcap C_2$, $C_1 \sqcup C_2$ and $\exists R.C$, the correctness of their interpretations follows directly from Definition 1 and the semantics of $\mathcal{ALCH}_{R^+}$ concept expressions given in Figure 1 above:

1. For concepts of the form $\neg C$, if $\neg C \in \mathcal{L}(s)$, then $C \notin \mathcal{L}(s)$, so $s \in \Delta^{\mathcal{I}} - C^{\mathcal{I}}$ and $\neg C$ is correctly interpreted.
2. For concepts of the form $C_1 \sqcap C_2$, if $C_1 \sqcap C_2 \in \mathcal{L}(s)$, then $C_1 \in \mathcal{L}(s)$ and $C_2 \in \mathcal{L}(s)$, so $s \in C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$ and $C_1 \sqcap C_2$ is correctly interpreted.
3. For concepts of the form $C_1 \sqcup C_2$, if $C_1 \sqcup C_2 \in \mathcal{L}(s)$, then $C_1 \in \mathcal{L}(s)$ or $C_2 \in \mathcal{L}(s)$, so $s \in C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$ and $C_1 \sqcup C_2$ is correctly interpreted.
4. For concepts of the form $\exists R.C$, if $\exists R.C \in \mathcal{L}(s)$, then there is some $t \in S$ such that $\langle s, t\rangle \in \mathcal{E}(R)$ and $C \in \mathcal{L}(t)$, so $s \in \{d \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(d) \cap C^{\mathcal{I}} \neq \emptyset\}$ and $\exists R.C$ is correctly interpreted.

For concepts of the form $\forall R.C$, the correctness of their interpretations follows from Definition 1, and the semantics of $\mathcal{ALCH}_{R^+}$ concept expressions and axioms:

1. if $\forall R.C \in \mathcal{L}(s)$ and $\langle s, t\rangle \in \mathcal{E}(R)$, then $C \in \mathcal{L}(t)$
2. if $\forall R.C \in \mathcal{L}(s)$, $\langle s, t\rangle \notin \mathcal{E}(R)$ and $\langle s, t\rangle \in R^{\mathcal{I}}$, then there exists a path $\langle s, u_1\rangle, \ldots, \langle u_n, t\rangle$ s.t. $n \geqslant 1$, $\{\langle s, u_1\rangle, \ldots, \langle u_n, t\rangle\} \subseteq \mathcal{E}(S)$, $S \in \mathbf{R}_+$ and $S \sqsubseteq R$. From property 6 of Definition 1, $\forall S.C \in \mathcal{L}(u_i)$ for all $1 \leqslant i \leqslant n$, and from property 4 of Definition 1, $C \in \mathcal{L}(t)$

so $s \in \{d \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(d) \subseteq C^{\mathcal{I}}\}$ and $\forall R.C$ is correctly interpreted.

Finally, from Definition 1, there is some individual $s \in \mathbf{S}$ such that $D \in \mathcal{L}(s)$, so $s \in D^{\mathcal{I}}$ and $D^{\mathcal{I}} \neq \emptyset$.

For the converse, if $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is a model of D, then a tableau $T = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ for D can be defined as:

$$\begin{aligned} S &= \Delta^{\mathcal{I}} \\ \mathcal{E}(R) &= R^{\mathcal{I}} \\ \mathcal{L}(s) &= \{C \in sub(D) \mid s \in C^{\mathcal{I}}\} \end{aligned}$$

It only remains to demonstrate that T is a tableau for D:

1. T satisfies properties 1–5 in Definition 1 as a direct consequence of the semantics of the $\neg C$, $C_1 \sqcap C_2$, $C_1 \sqcup C_2$, $\forall R.C$ and $\exists R.C$ concept expressions.
2. If $d \in (\forall R.C)^{\mathcal{I}}$, $\langle d, e\rangle \in S^{\mathcal{I}}$, $S \in \mathbf{R}_+$ and $S \sqsubseteq R$, then $e \in (\forall S.C)^{\mathcal{I}}$ unless there is some f such that $\langle e, f\rangle \in S^{\mathcal{I}}$ and $f \notin C^{\mathcal{I}}$. However, if $\langle d, e\rangle \in S^{\mathcal{I}}$, $\langle e, f\rangle \in S^{\mathcal{I}}$, $S \in \mathbf{R}_+$ and $S \sqsubseteq R$, then from the semantics of role axioms given in Figure 2, $S \in \mathbf{R}_+ \rightarrow \langle d, f\rangle \in S^{\mathcal{I}}$, $S \sqsubseteq R \rightarrow \langle d, f\rangle \in R^{\mathcal{I}}$ and $d \notin (\forall R.C)^{\mathcal{I}}$. T therefore satisfies property 6 in Definition 1.
3. T satisfies property 7 in Definition 1 as a direct consequence of the semantics of role inclusion axioms given in Figure 2. ■

3.2 CONSTRUCTING AN $\mathcal{ALCH}_{R^+}$ TABLEAU

The algorithm builds a tree where each node x of the tree is labelled with a set $\mathcal{L}(x) \subseteq sub(D)$ and may, in addition, be marked *satisfiable*. The tree is expanded either by extending $\mathcal{L}(x)$ for some leaf node x or by adding new leaf nodes. For a node x, $\mathcal{L}(x)$ is said to contain a *clash* if $\bot \subseteq \mathcal{L}(x)$ or, for some concept C, $\{C, \neg C\} \subseteq \mathcal{L}(x)$. $\mathcal{L}(x)$ is called a *pre-tableau* if it is clash-free and contains no unexpanded conjunction or disjunction concepts. Note that $\emptyset$ is a pre-tableau.

Edges of the tree are either labelled $\sqcup$ or labelled R for some role name R occurring in $sub(D)$. Edges labelled $\sqcup$ are added when expanding $C_1 \sqcup C_2$ concepts in $\mathcal{L}(x)$, and are the mechanism whereby the algorithm searches possible alternative expansions. Edges labelled with a role name R are added when expanding $\exists R.C$ terms in $\mathcal{L}(x)$, and correspond to relationships between pairs of individuals.

A node y is called an *R-successor* of a node x if there is an edge $\langle x, y\rangle$ labelled R; y is called a $\sqcup$-*successor* of x if there is a path, consisting of $\sqcup$-labelled edges, from x to y. A node x is an *ancestor* of a node y if there is a path from x to y regardless of the labelling of the edges. Note that both the $\sqcup$-successor and ancestor relations are reflexive as nodes are connected to themselves by the empty path.

The algorithm initialises a tree $\mathbf{T}$ to contain a single node x_0, called the *root* node, with $\mathcal{L}(x_0) = \{D\}$. $\mathbf{T}$ is then expanded by repeatedly applying the rules from Figure 3 until either the root node is marked *satisfiable* or none of the rules is applicable. Note that the second condition in each rule prevents multiple applications of the rule to the same concept expression(s), while blocking is performed by part b of the $\exists$-rule.

If the algorithm constructs a tree in which the root node is marked *satisfiable*, then it returns *satisfiable*; from this tree a tableau for D can trivially be constructed. If the algorithm terminates without marking the root node *satisfiable*, then it returns *unsatisfiable*.

3.3 SOUNDNESS AND COMPLETENESS

From Lemma 1, the soundness and completeness of the algorithm can be demonstrated by proving that, for an $\mathcal{ALCH}_{R^+}$-concept D, it always terminates and that it returns *satisfiable* iff there exists a tableau for D.

Lemma 2 *For each $\mathcal{ALCH}_{R^+}$-concept D, the tableau construction terminates.*

$\sqcap$-rule:	if 1.	x is a leaf of $\mathbf{T}$, $\mathcal{L}(x)$ is clash-free, $C_1 \sqcap C_2 \in \mathcal{L}(x)$
	2.	$\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$
	then	$\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{C_1, C_2\}$
$\sqcup$-rule:	if 1.	x is a leaf of $\mathbf{T}$, $\mathcal{L}(x)$ is clash-free, $C_1 \sqcup C_2 \in \mathcal{L}(x)$
	2.	$\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$
	then	create two $\sqcup$-successors y, z of x with: $\mathcal{L}(y) = \mathcal{L}(x) \cup \{C_1\}$, $\mathcal{L}(z) = \mathcal{L}(x) \cup \{C_2\}$
$\exists$-rule:	if 1.	$\mathcal{L}(x)$ is a pre-tableau, there is a concept of the form $\exists R.C$ in $\mathcal{L}(x)$
	2.	x is a leaf of $\mathbf{T}$
	then	for each $\exists R.C \in \mathcal{L}(x)$ do:
	a.	$\ell_{Rx} := \{C\} \cup \{D \mid \forall S.D \in \mathcal{L}(x) \text{ and } R \sqsubseteq S\} \cup \{\forall S.D \mid \forall P.D \in \mathcal{L}(x),\ S \in \mathbf{R}_+,\ S \sqsubseteq P \text{ and } R \sqsubseteq S\}$
	b.	if for some ancestor w of x, $\ell_{Rx} \subseteq \mathcal{L}(w)$ then create an R-successor y of x with $\mathcal{L}(y) = \emptyset$
	c.	otherwise create an R-successor y of x with $\mathcal{L}(y) = \ell_{Rx}$
SAT-rule:	if 1.	x is a node of $\mathbf{T}$, and either:
	a.	$\mathcal{L}(x)$ is a pre-tableau containing no concepts of the form $\exists R.C$
	b.	$\mathcal{L}(x)$ is a pre-tableau which has successors, and all successors of x are marked *satisfiable*
	c.	$\mathcal{L}(x)$ is not a pre-tableau and some $\sqcup$-successor of x is marked *satisfiable*
	2.	x is not marked *satisfiable*
	then	mark x *satisfiable*

Figure 3: Tableaux Expansion Rules for $\mathcal{ALCH}_{R+}$

Proof: Let $m = |sub(D)|$. As nodes are labelled with subsets of $sub(D)$, $|\mathcal{L}(x)| \leq m$ for all nodes x. For any node x the $\sqcap$-rule can therefore be applied at most m times. The size of any sub-trees is also limited by m: the $\sqcup$-rule can also be applied at most m times along a $\sqcup$-labelled path and the $\exists$-rule can be applied at most 2^m times along any path before there must be some ancestor y s.t. $\ell_{Rx} \subseteq \mathcal{L}(y)$ for any R.

Lemma 3 *For an $\mathcal{ALCH}_{R+}$-concept D, there exists a tableau for D iff the tableau construction returns* satisfiable.

Proof: For the *if* direction (the algorithm returns *satisfiable*), let $\mathbf{T}$ be the tree constructed by the tableaux algorithm for D. A tableau $T = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ can be defined with:

$$\begin{array}{rcl} \mathbf{S} & = & \{x \mid x \text{ is a node in } \mathbf{T},\ x \text{ is marked \textit{satisfiable} and } \mathcal{L}(x) \text{ is a non-empty pre-tableau.}\} \\ \mathcal{E}(R) & = & \{\langle x, y\rangle \in \mathbf{S} \times \mathbf{S} \mid \textit{either } y \text{ is a } \sqcup\text{-successor of an } R\text{-successor of } x;\ \textit{or } x \text{ has an } R\text{-successor } z \text{ with } \mathcal{L}(z) = \emptyset,\ y \text{ is an ancestor of } x \text{ and } \ell_{Rx} \subseteq \mathcal{L}(y);\ \textit{or } \text{for some role S, } \langle x, y\rangle \in \mathcal{E}(S) \text{ and } S \sqsubseteq R\} \end{array}$$

and it can be shown that T is a tableau for D:

1. $D \in \mathcal{L}(x)$ for the root x_0 of $\mathbf{T}$ and for all $\sqcup$-successors of x_0. As x_0 is marked *satisfiable* one of these must be a non-empty pre-tableau marked *satisfiable*, so $D \in \mathcal{L}(s)$ for some $s \in \mathbf{S}$.

2. Properties 1–3 of Definition 1 are satisfied because each $x \in \mathbf{S}$ is a pre-tableau.

3. Property 4 in Definition 1 is satisfied because $\{C \mid \forall R.C \in \mathcal{L}(x)\} \subseteq \ell_{Rx}$ and $\ell_{Rx} \subseteq \mathcal{L}(y)$ for all y with $\langle x, y\rangle \in \mathcal{E}(R)$.

4. Property 5 in Definition 1 is satisfied by the $\exists$-rule which, for all $x \in \mathbf{S}$, creates for each $\exists R.C \in \mathcal{L}(x)$ a new R-successor y with either:
   (a) $C \in \mathcal{L}(y)$ *or*
   (b) $\mathcal{L}(y) = \emptyset$, $C \in \ell_{Rx}$ and $\ell_{Rx} \subseteq \mathcal{L}(z)$ for some ancestor z of x.
5. Property 6 in Definition 1 is satisfied because $\{\forall S.C \mid \forall R.C \in \mathcal{L}(x), S \in \mathbf{R}_+ \text{ and } S \sqsubseteq R\} \subseteq \ell_{Sx}$ and $\ell_{Sx} \subseteq \mathcal{L}(y)$ for all y with $\langle x, y\rangle \in \mathcal{E}(S)$.
6. Property 7 in Definition 1 is satisfied because $\langle x, y\rangle \in \mathcal{E}(S)$ for all $\langle x, y\rangle \in \mathcal{E}(R)$ and $R \sqsubseteq S$.

For the converse (the algorithm returns *unsatisfiable*), it can be shown by induction on $h(x)$, the height of the sub-tree below x, that if x is not marked *satisfiable* then the concept $X = \sqcap_{C \in \mathcal{L}(x)} C$ is not satisfiable:

1. If $h(x) = 0$ (x is a leaf) and x is not marked *satisfiable*, then $\mathcal{L}(x)$ contains a clash and X is clearly unsatisfiable.
2. If $h(x) > 0$, $\mathcal{L}(x)$ is not a pre-tableau and x is not marked *satisfiable*, then none of its $\sqcup$-successors is marked *satisfiable*; hence $C_1 \sqcup C_2 \in \mathcal{L}(x)$ and neither y with $\mathcal{L}(y) = \mathcal{L}(x) \cup \{C_1\}$ nor z with $\mathcal{L}(z) = \mathcal{L}(x) \cup \{C_2\}$ is marked *satisfiable*. It follows by induction that X is not satisfiable.
3. If $h(x) > 0$, $\mathcal{L}(x)$ is a pre-tableau and x is not marked *satisfiable*, then there is some R-successor of x which is not marked *satisfiable* and it follows by induction, and the semantics of value restriction concept expressions ($\forall R.C$), that X is not satisfiable. ∎

4 EXTENDING $\mathcal{ALCH}_{R^+}$ WITH ATTRIBUTES

$\mathcal{ALCH}_{R^+}$ can be extended with limited support for attributes (functional/feature roles) to give $\mathcal{ALCH}f_{R^+}$. Unlike $\mathcal{ALCF}$ (Hollunder and Nutt 1990), $\mathcal{ALCH}f_{R^+}$ does not include support for attribute value map concept forming operators, but it only requires a minor extension to the $\mathcal{ALCH}_{R^+}$ algorithm.

$\mathcal{ALCH}f_{R^+}$ extends the syntax of $\mathcal{ALCH}_{R^+}$ by allowing axioms of the form $A \in \mathbf{F}$ to appear in terminologies, where A is a role name and F is the set of functional role names, or attributes. As well as being correct for $\mathcal{ALCH}_{R^+}$ concept expressions, an $\mathcal{ALCH}f_{R^+}$ interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ must satisfy the additional condition that, for all $A \in \mathbf{F}$, $A^{\mathcal{I}}$ is a single valued partial function, $A^{\mathcal{I}} : dom\ A^{\mathcal{I}} \longrightarrow \Delta^{\mathcal{I}}$.

The $\exists$-rule in the $\mathcal{ALCH}_{R^+}$ tree construction algorithm can be extended to deal with attributes in $\mathcal{ALCH}f_{R^+}$. Expressions of the form $\exists R.C$, where R is a role, are dealt with exactly as before, but expressions of the form $\exists A.C$, where A is an attribute, require special treatment. The extended rule treats attributes in a similar way to roles: $\exists A.C$ expressions in the label of a pre-tableau node x will lead to the creation of new $\mathbf{A}$-successor nodes y_i and labelled edges $\langle x, y_i\rangle$. However, it may group together multiple $\exists A.C$ expressions in x's label to create a single $\mathbf{A}$-successor y, labeling the edge $\langle x, y\rangle$ with a set of attribute names $\mathbf{A}$.

Multiple $\exists A.C$ expressions must be grouped together when, in the model represented by the tree, the $A^{\mathcal{I}}(x)$ are constrained to be the same individual, for example when there are multiple $\exists A.C$ expressions containing the same attribute A. The interaction between attributes and the role hierarchy means that for two expressions $\exists A.C_1 \in \mathcal{L}(x)$ and $\exists B.C_2 \in \mathcal{L}(x)$, where A and B are attributes, $A^{\mathcal{I}}(x)$ and $B^{\mathcal{I}}(x)$ are also constrained to be the same individual when $A \sqsubseteq B$ (because $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$) or $B \sqsubseteq A$ (because $B^{\mathcal{I}} \subseteq A^{\mathcal{I}}$). We will say that an attribute B is *directly-constrained* by an attribute A in $\mathcal{L}(x)$ if ($\exists A.C \in \mathcal{L}(x)$ and $A \sqsubseteq B$) or ($\exists B.C \in \mathcal{L}(x)$ and $B \sqsubseteq A$), and we will say that an attribute B is *constrained* by an attribute A in $\mathcal{L}(x)$ if B is directly-constrained by A in $\mathcal{L}(x)$ or if, for some attribute A', A' is directly-constrained by A in $\mathcal{L}(x)$ and B is constrained by A' in $\mathcal{L}(x)$. For an attribute B and a node x, the set of attributes which are constrained by B in $\mathcal{L}(x)$ will be denoted $\mathbf{A}_{Bx}$, where $\mathbf{A}_{Bx} = \{A \in \mathbf{F} \mid A \text{ is constrained by } B \text{ in } \mathcal{L}(x)\}$. The extended $\exists$-rule for $\mathcal{ALCH}f_{R^+}$ is shown in Figure 4.

5 GENERAL TERMINOLOGIES

The algorithm described in Section 3 tests the satisfiability of a concept expression D w.r.t. a terminology which does not contain concept axioms, but it can also be used to test satisfiability w.r.t. an arbitrary terminology $\mathcal{T}$. If $\mathcal{T}$ is a *restricted* terminology, one which contains only acyclic concept definitions,[3] this can be achieved simply by *unfolding* D—using the definitions in $\mathcal{T}$ to expand concept names in D until they are all atomic primitives. The $\mathcal{ALCH}f_{R^+}$ algorithm can, however, also be used to test the satisfiability of a concept expression with respect to a *general* terminology, one which may contain both cycles and general concept inclusion axioms (GCIs). A GCI is an axiom of the form $C \sqsubseteq D$ where C is an arbitrary concept expression.

[3]A concept definition is an axiom of the form CN $\sqsubseteq C$ or CN $\doteq C$, where CN is a concept name which appears only once on the left hand side of such an axiom. Concept definitions are acyclic if C does not refer to CN, either directly or indirectly. Concepts defined by an axiom of the form CN $\sqsubseteq C$ are called primitive while those defined by an axiom of the form CN $\doteq C$ are called non-primitive.

```
if 1.   $\mathcal{L}(x)$ is a pre-tableau, there is a concept of the form $\exists R.C$ in $\mathcal{L}(x)$
   2.   $x$ is a leaf of $\mathbf{T}$
then    for each $\exists R.C \in \mathcal{L}(x)$ where $R \notin \mathbf{F}$ do:
          a.  $\ell_{Rx} := \{C\} \cup \{D \mid \forall S.D \in \mathcal{L}(x)$ and $R \sqsubseteq S\}$
                        $\cup \{\forall S.D \mid \forall P.D \in \mathcal{L}(x),\ S \in \mathbf{R}_+,\ S \sqsubseteq P$ and $R \sqsubseteq S\}$
          b.  if for some ancestor $w$ of $x$, $\ell_{Rx} \subseteq \mathcal{L}(w)$
              then create an $R$-successor $y$ of $x$ with $\mathcal{L}(y) = \emptyset$
          c.  otherwise create an $R$-successor $y$ of $x$ with $\mathcal{L}(y) = \ell_{Rx}$
and     for each $\exists A.D \in \mathcal{L}(x)$ where $A \in \mathbf{F}$ do:
          a.  if for some $\mathbf{A}$-successor $y$ of $x$, $A \in \mathbf{A}$ then do nothing.
          b.  otherwise
                i.    $\mathbf{A} := \mathbf{A}_{Ax}$
                ii.   $\ell_{Ax} := \bigcup_{B \in \mathbf{A}}$  $(\{C \mid \exists B.C \in \mathcal{L}(x)\} \cup$
                                    $\{C \mid \forall S.C \in \mathcal{L}(x)$ and $B \sqsubseteq S\} \cup$
                                    $\{\forall S.C \mid \forall P.C \in \mathcal{L}(x),\ S \in \mathbf{R}_+,\ S \sqsubseteq P$ and $B \sqsubseteq S\})$
                iii.  if for some ancestor $w$ of $x$, $\ell_{Ax} \subseteq \mathcal{L}(w)$
                      then create an $\mathbf{A}$-successor $y$ of $x$ with $\mathcal{L}(y) = \emptyset$
                iv.   otherwise create an $\mathbf{A}$-successor $y$ of $x$ with $\mathcal{L}(y) = \ell_{Ax}$
```

Figure 4: Extended $\exists$-rule for $\mathcal{ALCH}f_{R+}$

An axiom $C \doteq D$ is equivalent to two GCIs, $C \sqsubseteq D$ and $D \sqsubseteq C$, so we can, without loss of generality, restrict our attention to GCIs.

A procedure called *internalisation* (Baader 1990) can be used to test the satisfiability of a concept expression D with respect to a terminology $\mathcal{T}$ containing an arbitrary set of GCIs $\{A_1 \sqsubseteq B_1, \ldots, A_n \sqsubseteq B_n\}$. Internalisation works by testing the satisfiability of $D \sqcap \mathcal{M} \sqcap \forall U.\mathcal{M}$, where $\mathcal{M}$ is a concept expression formed from the GCIs, $\mathcal{M} \doteq (B_1 \sqcup \neg A_1) \sqcap \ldots \sqcap (B_n \sqcup \neg A_n)$, and U is a specially defined transitive role which subsumes all the other roles which occur in $\mathcal{T}$. The properties of U ensure that, in any model constructed by the tableaux algorithm, every individual satisfies $\mathcal{M}$, and thus satisfies each of the GCIs in $\mathcal{T}$.

Assuming descriptive rather than fixed point semantics (Nebel 1990), terminological cycles can easily be dealt with by treating all concept axioms as GCIs (Buchheit et al. 1993). However, this method is highly inefficient because reasoning with GCIs introduces large numbers of disjunctions and is thus very costly. Terminological cycles can be dealt with in a much more efficient manner by using *lazy unfolding*: using the definitions in $\mathcal{T}$ to expand concept names in D, but only as required by the progress of the tableau expansion (Baader et al. 1992).

When building a tree $\mathbf{T}$, lazy unfolding ensures that if the terminology $\mathcal{T}$ contains a primitive definition axiom CN $\sqsubseteq C$, then for any node x in $\mathbf{T}$, CN $\in \mathcal{L}(x) \Rightarrow C \in \mathcal{L}(x)$. Therefore, in the model represented by $\mathbf{T}$, $\mathsf{CN}^{\mathcal{I}} \subseteq C^{\mathcal{I}}$ and the axiom is satisfied. If C refers either directly or indirectly to CN, termination of the tree construction algorithm is still guaranteed because of blocking—most implemented DLs are unable to deal with terminological cycles because they have no blocking mechanism and could not guarantee termination.

Lazy unfolding also takes care of non-primitive definition axioms CN $\doteq C \in \mathcal{T}$, provided that C can be unfolded so that it contains only primitive concepts, as any primitive interpretation (an assignment of values to the interpretations of primitive concepts) will lead, via the semantics, to an interpretation for CN such that $\mathsf{CN}^{\mathcal{I}} = C^{\mathcal{I}}$. However, if C cannot be unfolded so that it contains only primitive concepts, then it cannot be guaranteed that a model constructed by the algorithm satisfies $\mathcal{T}$. For example, if $\mathcal{T} = \{\mathsf{CN}_1 \sqsubseteq \top, \mathsf{CN}_2 \doteq \neg\mathsf{CN}_2\}$, then $\mathcal{T}$ is obviously unsatisfiable (it only has a model with an empty domain). Testing the satisfiability of CN_1 would, however, cause the algorithm to build a tree representing a model where $\Delta^{\mathcal{I}} = \{x\}$ and $\mathsf{CN}_1^{\mathcal{I}} = \{x\}$.

This problem can be dealt with by checking each definition axiom CN $\doteq C \in \mathcal{T}$, and if C cannot be unfolded until it contains only primitive concepts, then transforming the axiom into a primitive definition CN $\sqsubseteq C$ and a GCI $C \sqsubseteq$ CN. The axiom $\mathsf{CN}_2 \doteq \neg\mathsf{CN}_2$ from the above example would thus be converted into the primitive definition $\mathsf{CN}_2 \sqsubseteq \neg\mathsf{CN}_2$ and the GCI $\neg\mathsf{CN}_2 \sqsubseteq \mathsf{CN}_2$. The GCI would lead to CN_2 being added to every node label, and the unfolding of CN_2 would then add $\neg\mathsf{CN}_2$, causing an immediate clash.

6 THE FaCT SYSTEM

The FaCT system is a terminological classifier (TBox) which has been developed as a testbed for a highly optimised implementation of the $\mathcal{ALCH}f_{R+}$ satisfiability testing algorithm, and to evaluate its empirical tractability. FaCT reasons about concept, role and attribute descriptions, and maintains a concept hierarchy based on the subsumption relation. The algorithm is used for subsumption testing in the usual way: C subsumes D iff $D \sqcap \neg C$ is not satisfiable. Correspondences between modal and description logics (Schild 1991) mean that FaCT can also be used as a theorem prover for the propositional modal logics $\mathbf{K}_{(\mathbf{m})}$, $\mathbf{KT}_{(\mathbf{m})}$, $\mathbf{K4}_{(\mathbf{m})}$ and $\mathrm{S4}_{(\mathbf{m})}$.

6.1 OPTIMISATION TECHNIQUES

A naive implementation of the algorithm would be of limited value in realistic applications: when trying to classify the GALEN medical terminology ontology, for example, single satisfiability problems were encountered which the unoptimised algorithm had failed to solve after 100 hours of CPU time. To improve the performance of the algorithm, a range of optimisations have been employed (Horrocks 1997). These include:

- Lexical normalisation and encoding of concept expressions—a technique which takes the hierarchical structure of terminologies to its logical conclusion by lexically normalising and encoding all concept expressions and, recursively, their sub-expressions. In this form, concept expressions consist only of (possibly negated) concept names, conjunctions ($C_1 \sqcap \ldots \sqcap C_n$) and value restrictions ($\forall R.C$): expressions of the form $\exists R.C$ are transformed into $\neg(\forall R.\neg C)$ and expressions of the form ($C_1 \sqcup \ldots \sqcup C_n$) are transformed into $\neg(\neg C_1 \sqcap \ldots \sqcap \neg C_n)$. In addition, the sub-expressions forming conjunctions are sorted and any duplicates eliminated. The normalisation process also identifies and simplifies sub-expressions which are obviously satisfiable (e.g., $\forall R.\top$) or obviously unsatisfiable (e.g., $(C \sqcap \neg C \sqcap \ldots)$), replacing them with $\top$ or $\bot$ respectively: in extreme cases (when the whole expression simplifies to $\top$ or $\bot$) the need for a tableau expansion can be completely eliminated.

  The encoding process gives a unique identifier to each lexically distinct concept expressions which, in conjunction with lazy unfolding and the retention of unfolded identifiers, facilitates early clash detection when an identifier and its negation occur in the same node label (Baader et al. 1992).

- Absorption—a technique which eliminates GCIs from a terminology by absorbing them into primitive concept definition axioms. For example, if a terminology contains the axiom $\mathsf{P} \sqsubseteq C$ and the GCI $\mathsf{P} \sqcap D_1 \sqsubseteq D_2$, the GCI can be eliminated from the terminology by absorbing it into the axiom to give $\mathsf{P} \sqsubseteq C \sqcap (D_2 \sqcup \neg D_1)$.

  Although absorption adds a disjunction to the primitive concept definition axiom, it is much more efficient than reasoning w.r.t. the GCI, which would require the disjunction $D_2 \sqcup \neg(\mathsf{P} \sqcap D_1)$ to be added to the label of every node. In effect, absorption restricts the application of this disjunction to nodes where it is really required.

- Semantic branching—a search technique adapted from the Davis-Putnam-Logemann-Loveland procedure (DPL) commonly use to solve propositional satisfiability (SAT) problems (Davis et al. 1962). Semantic branching works by selecting a concept C from one of the unexpanded disjunctions in the label of a node x and searching $\mathcal{L}(x) \cup \{C\}$ and $\mathcal{L}(x) \cup \{\neg C\}$. Wasted search is avoided because the two branches are strictly disjoint. For example, if $\{C \sqcup D, C \sqcup E\} \subseteq \mathcal{L}(x)$ and $\mathcal{L}(x) \cup \{C\}$ is found to be unsatisfiable, then $\neg C$ is added to $\mathcal{L}(x)$ and a second, possibly costly, evaluation of the unsatisfiability of $\mathcal{L}(x) \cup \{C\}$ is avoided. A similar technique is also used in the KSAT modal $\mathbf{K}_{(\mathbf{m})}$ (equivalently $\mathcal{ALC}$ (Schild 1991)) satisfiability testing algorithm (Giunchiglia and Sebastiani 1996).

- Dependency directed backtracking—a technique adapted from constraint satisfiability problem solving (Baker 1995) which addresses the problem of thrashing (large amounts of unproductive backtracking search) caused by inherent unsatisfiability concealed in sub-problems. Backjumping labels concept expressions with a dependency set indicating the branch points on which they depend. When a clash is discovered, the dependency sets can be used to identify the most recent branch point where exploring the other branch might alleviate the cause of the clash. The algorithm can then jump back over intervening branch points *without* exploring alternative branches. A similar technique was used in the HARP theorem prover (Oppacher and Suen 1988).

- Caching and re-using partial models—a technique which takes advantage of the repetitive structure of the satisfiability problems generated during terminological classification by using cached partial tableaux to demonstrate "obvious" satisfiability. For example, the satisfiability of the concept expression $C \sqcap \neg D$ (and thus the non-subsumption $C \not\sqsubseteq D$) can be demonstrated by showing that tableaux for C and $\neg D$ joined at their root nodes result in a valid tableau for $C \sqcap \neg D$.

6.2 EMPIRICAL EVALUATION

The performance of the FaCT system has been evaluated using a variety of empirical testing procedures (Horrocks 1997). When assessing the results of these tests it is important to note the the current system is an experimental prototype written in Common Lisp, and that very little consideration has been given to low-level efficiency issues. The tests have been performed using Allegro CL 4.3 (compiled) on a Sun SPARCstation 20/61 equipped with a 60MHz superSPARC processor, a 1Mbyte off-chip cache and 128Mbytes of RAM. The FaCT system and test KBs are available from the author's home page.

To demonstrate the feasibility of using FaCT with a large, realistic KB, it has been used to classify an $\mathcal{ALCH}f_{R^+}$ KB representing the GALEN medical terminology ontology. The KB used in the tests (which has since been extend as part of the ongoing GALEN project) contains 2,740 concepts, 699 of which are non-primitive, 413 roles, 26 of which are transitive, and 1,214 GCIs. Using the optimised algorithm, FaCT is able to classify the KB in ≈379s of CPU time, performing a total of 122,695 subsumption tests at an average of 0.003s per test. FaCT's performance contrasts with that of the KRIS system (Baader and Hollunder 1991) which had only classified a small proportion (≈10%) of a simplified version of the KB (with cycles and transitive roles eliminated) after 100 hours of CPU time.

FaCT also performs well as a modal logic theorem prover: Table 1 compares FaCT with CRACK (Bresciani et al. 1995), KSAT and KRIS using a suite of benchmark formulae for modal **K** (Heuerding and Schwendimann 1996). The tests use 9 classes of formula (*k_branch*, *k_d4*, etc.) in both provable (p) and non-provable (n) forms.[4] For each type of formula, 21 examples of exponentially increasing difficulty are provided, and the table shows the number of the largest formula which each system was able to solve within 100 seconds of CPU time (21 indicates that the hardest problem was solved in less than 100s).

FaCT significantly outperformed all the other systems, and in many cases also exhibited a completely different qualitative performance. For example, with *k_dum_p* formulae (see Figure 5) the other systems all showed an exponential increase in solution times with increasing formula size, whereas the times taken by FaCT increased very little for larger formulae (and FaCT was already 2,000 times faster for the largest formula solved by another system).

[4]Note that a formula is proved by demonstrating the unsatisfiability of its negation.

Table 1: Modal **K** Theorem Proving

	FaCT		Crack		KSAT		Kris	
Test	p	n	p	n	p	n	p	n
k_branch	6	4	2	1	8	8	3	3
k_d4	21	8	2	3	8	5	8	6
k_dum	21	21	3	21	11	21	15	21
k_grz	21	21	1	21	17	21	13	21
k_lin	21	21	5	2	21	3	6	9
k_path	7	6	2	6	4	8	3	11
k_ph	6	7	2	3	5	5	4	5
k_poly	21	21	21	21	13	12	11	21
k_t4p	21	21	1	1	10	18	7	5

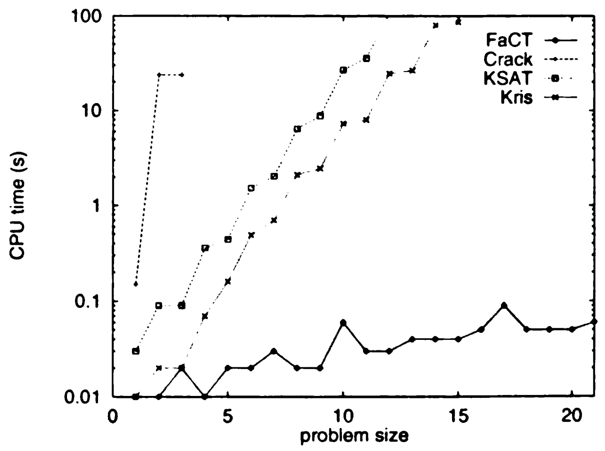


Figure 5: Solution Times for *k_dum_p* Formulae

7 DISCUSSION

This paper describes a sound and complete satisfiability testing algorithm for a relatively expressive DL, one which can reason with respect to a general terminology and a primitive hierarchy of transitive and non-transitive roles. In contrast to most other theoretical presentations, a practical system which uses an (optimised) implementation of the algorithm is also described. The FaCT system has been used to investigate the practicability of using the algorithm for subsumption reasoning, and results so far suggest that in spite of the logic's worst-case intractability the algorithm can provide acceptable performance in realistic applications. FaCT has also been shown to perform well when used as a propositional modal logic theorem prover, and detailed results from these experiments will be the subject of a future paper.

Although the "nice" properties of transitive roles, as opposed to a transitive closure operator (see Section 2), made it simple to implement and optimise the algorithm, many of the techniques investigated could be used with other tableaux satisfiability testing algorithms, and should become standard in future tableaux based DL implementations. Normalisation, encoding and absorption can, for example, be performed as pre-processing steps, and could be used with any DL regardless of its subsumption testing algorithm, although integrating normalisation and encoding with the classifier is preferable in order to avoid the overhead of classifying new concepts generated by the encoding process. The results obtained with FaCT suggest that some of the very expressive DLs for which tableaux algorithms are now available may also be usable in realistic applications, and work is already underway to produce an optimised implementation of such an algorithm.

Acknowledgements

I would like to thank Graham Gough, Alan Rector, Carole Goble and Richard Banach for their help and support. Ulrike Sattler made significant contributions, both technical and inspirational. Thanks are also due to Franz Baader, Sean Bechhofer, Enrico Franconi, Fausto Giunchiglia, Peter Patel-Schneider, Ian Pratt, Dominik Schoop and Roberto Sebastiani for invaluable discussions, suggestions and feedback. Finally, thanks again to Enrico Franconi for his recent hospitality at ITC-IRST, Trento, Italy were some of the work presented was carried out.

The work has been supported by the Engineering and Physical Sciences Research Council, the "Integration and Access to Heterogeneous Databases" Italian Space Agency (ASI) project, and by the "Foundations of Data Warehouse Quality" (DWQ) European ESPRIT Long Term Research Project 22469.

References

L. C. Aiello, J. Doyle, and S. Shapiro, editors. *Principals of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR'96)*. Morgan Kaufmann Publishers, San Francisco, CA, 1996.

A. Artale, N. Guarino, E. Franconi, and L. Pazzi. Part-whole relations in object-centered systems: an overview. *Data and Knowledge Engineering*, 20:347–383, 1996.

F. Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. Research Report RR-90-13, Deutsches Forschungszentrum für Künstliche Intelligenz GmbH (DFKI), 1990.

F. Baader, M. Buchheit, M.A. Jeusfeld, and W. Nutt, editors. *Reasoning about structured objects: knowledge representation meets databases. Proceedings of the 3rd Workshop KRDB'96*, 1996.

F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H.-J. Profitlich. An empirical analysis of optimization techniques for terminological representation systems or: Making KRIS get a move on. In B. Nebel, C. Rich, and W. Swartout, editors, *Principals of Knowledge Representation and Reasoning: Proceedings of the Third International Conference (KR'92)*, pages 270–281. Morgan-Kaufmann Publishers, San Francisco, CA, 1992. Also available as DFKI RR-93-03.

F. Baader, H.-J. Heinsohn, B. Hollunder, J. Muller, B. Nebel, W. Nutt, and H.-J. Profitlich. Terminological knowledge representation: A proposal for a terminological logic. Technical Memo TM-90-04, Deutsches Forschungszentrum für Künstliche Intelligenz GmbH (DFKI), 1991.

F. Baader and B. Hollunder. A terminological knowledge representation system with complete inference algorithms. In *Processing declarative knowledge: International workshop PDK'91*, number 567 in Lecture Notes in Artificial Intelligence, pages 67–86, Berlin, 1991. Springer-Verlag.

A. B. Baker. *Intelligent Backtracking on Constraint Satisfaction Problems: Experimental and Theoretical Results*. PhD thesis, University of Oregon, 1995.

J. I. Berman, H. H. Moore IV, and J. R. Wright. CLASSIC and PROSE stories: Enabling technologies for knowledge based systems. *AT&T Technical Journal*, pages 69–78, 1994.

A. Borgida. Description logics are not just for the flightless-birds: A new look at the utility and foundations of description logics. Technical Report DCS-TR-295, New Brunswick Department of Computer Science, Rutgers University, 1992.

P. Bresciani, E. Franconi, and S. Tessaris. Implementing and testing expressive description logics: a preliminary report. In Ellis et al. (1995), pages 28–39.

M. Buchheit, F. M. Donini, and A. Schaerf. Decidable reasoning in terminological knowledge representation systems. *Journal of Artificial Intelligence Research*, 1:109–138, 1993.

M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Communications of the ACM*, 5:394–397, 1962.

F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. Research Report RR-95-07, Deutsches Forschungszentrum für Künstliche Intelligenz GmbH (DFKI), 1995.

J. Doyle and R. Patil. Two theses of knowledge representation: Language restrictions, taxonomic classification, and the utility of representation services. *Artificial Intelligence*, 48:261–297, 1991.

Gerard Ellis, Robert A. Levinson, Andrew Fall, and Veronica Dahl, editors. *Knowledge Retrieval, Use and Stor-*

age for Efficiency: Proceedings of the First International KRUSE Symposium, 1995.

G. De Giacomo and M. Lenzerini. TBox and ABox reasoning in expressive description logics. In Aiello et al. (1996), pages 316–327.

F. Giunchiglia and R. Sebastiani. A SAT-based decision procedure for $\mathcal{ALC}$. In Aiello et al. (1996), pages 304–314.

C. A. Goble, C. Haul, and S. Bechhofer. Describing and classifying multimedia using the description logic GRAIL. In *Proceedings of IS&T/SPIE, vol 2670, Storage and Retrieval for Still Image and Video Databases* {*IV*},' pages 132–143, San Jose, California, USA, 1996.

R. V. Guha and D. B. Lenat. Enabling agents to work together. *Communications of the ACM*, 37(7):127–142, 1994.

J. Heinsohn, D. Kudenko, B. Nebel, and H.-J. Profitlich. An empirical analysis of terminological representation systems. *Artificial Intelligence*, 68:367–397, 1994.

A. Heuerding and S. Schwendimann. A benchmark method for the propositional modal logics k, kt, s4. Technical report IAM-96-015, University of Bern, Switzerland, 1996.

B. Hollunder and W. Nutt. Subsumption algorithms for concept languages. Research Report RR-90-04, Deutsches Forschungszentrum für Künstliche Intelligenz GmbH (DFKI), 1990.

I. Horrocks. *Optimising Tableaux Decision Procedures for Description Logics*. PhD thesis, University of Manchester, 1997.

I. Horrocks, A. Rector, and C. Goble. A description logic based schema for the classification of medical data. In Baader et al. (1996), pages 24–28.

P. Hors. Description logics to specify the part–whole relation. In *Proceedings of the ECAI'94 Workshop on Parts and Wholes: Conceptual Part–Whole Relations and Formal Mereology*, pages 103–109, 1994.

U. Küssner. Applying DL in automatic dialoge interpreting. In M.-C. Rousset, R. Brachman, F. Donini, E. Franconi, I. Horrocks, and A. Levy, editors, *Collected Papers from the International Description Logics Workshop (DL'97)*, pages 54–58, 1997.

A. Y. Levy and M.-C. Rousset. Using description logics to model and reason about views. In Baader et al. (1996), pages 48–49.

R. M. MacGregor. The evolving technology of classification-based knowledge representation systems. In J. F. Sowa, editor, *Principals of Semantic Networks: Explorations in the representation of knowledge*, chapter 13, pages 385–400. Morgan-Kaufmann, 1991.

E. Mays, R. Weida, R. Dionne, M. Laker, B. White, C. Liang, and F. J. Oles. Scalable and expressive medical terminologies. In *Proceedings of the 1996 AMAI Fall Symposium*, 1996.

B. Nebel. *Reasoning and Revision in Hybrid Representation Systems*. Number 422 in Lecture Notes in Artificial Intelligence. Springer-Verlag, Berlin, 1990.

W. A. Nowlan. *Structured methods of information management for medical records*. PhD thesis, University of Manchester, 1993.

F. Oppacher and E. Suen. HARP: A tableau-based theorem prover. *Journal of Automated Reasoning*, 4:69–100, 1988.

L. Padgham and P. Lambrix. A framework for part–of hierarchies in terminological logics. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Principals of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference (KR'94)*, pages 485–496. Morgan-Kaufmann Publishers, San Francisco, CA, 1994.

A. Rector, S. Bechhofer, C. A. Goble, I. Horrocks, W. A. Nowlan, and W. D. Solomon. The GRAIL concept modelling language for medical terminology. *Artificial Intelligence in Medicine*, 9:139–171, 1997.

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. To appear.

A. L. Rector, W A Nowlan, and A Glowinski. Goals for concept representation in the GALEN project. In *Proceedings of the 17th Annual Symposium on Computer Applications in Medical Care (SCAMC'93)*, pages 414–418, Washington DC, USA, 1993.

U. Sattler. A concept language for engineering applications with part–whole relations. In *Proceedings of the International Conference on Description Logics—DL'95*, pages 119–123, Roma, Italy, 1995.

U. Sattler. A concept language extended with different kinds of transitive roles. In G. Görz and S. Hölldobler, editors, *20. Deutsche Jahrestagung für Künstliche Intelligenz*, number 1137 in Lecture Notes in Artificial Intelligence, pages 333–345. Springer Verlag, 1996.

K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 466–471, 1991.

M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48:1–26, 1991.

P.-H. Speel, F. van Raalte, P. E. van der Vet, and N. J. I. Mars. Runtime and memory usage performance of description logics. In Ellis et al. (1995), pages 13–27.

W. A. Woods and J. G. Schmolze. The KL-ONE family. *Computers and Mathematics with Applications – Special Issue on Artificial Intelligence*, 23(2–5):133–177, 1992.

Invited Talks

How to Tailor Representations to Different Requirements
Abstract only

Katharina Morik
Universität Dortmund
Fachbereich Informatik, LS VIII
D-44221 Dortmund, Germany
morik@kirmes.informatik.uni-dortmund.de

The dream of a general purpose knowledge representation language has long been abandoned. However, most work on tailoring representation formalisms to particular needs has investigated ***deductive*** inference. The focus has been on inferential services for problem solving, e.g. classification or planning. The needs of the end user, be it a system (e.g., a natural language system or a robot) or a human user, determine the requirements for efficiency and expressiveness. If we focus on the knowledge engineer, additional requirements come into play: inspectability and revisability are major concerns in the process of knowledge acquisition. However, the underlying inference is still deductive. But the requirements on a knowledge representation formalism change when inductive inference is considered. A representation formalism with less expressive power may be harder to learn than a one with a higher expressive power. This means that complexity results for deductive reasoning cannot easily be transferred to inductive reasoning. Moreover, we frequently we encounter conflicting requirements for learning and problem solving. In this situation, asking for a representation that fulfils the requirements for both deductive and inductive inference, is akin to asking for a general purpose language. Instead, we design families of representations, where each family member is well suited for a particular set of requirements, and implement transformations between the representations.

In this talk, I discuss the representation family of Horn logic. Several restrictions of Horn logic have been investigated that ease learning. Three case studies illustrate how to tailor admissible languages. The first case study from a robotics application shows how a representation that is well suited for learning is transformed into an efficient deductive reasoner. The second case study exploits learning in order to enhance the understandibility and inspectability of a knowledge base under construction by a knowledge engineer. The third case study presents a tool that generates mappings from a relational database scheme to a Horn logic signature. Such mappings allow learning to take place directly from a relational database.

What Robots Can Do
Abstract only

Hector Levesque
Dept. of Computer Science
University of Toronto
Toronto, Ont. M5S 3H5, Canada
hector@cs.toronto.edu

We propose a definition of goal achievability: given a basic action theory describing an initial state of the world and some primitive actions available to a robot, including some actions which return sensing information, what goals can be achieved by the robot? The main technical result is a proof that a simple robot programming language is universal, in that any effectively achievable goal can be achieved by getting the robot to execute one of these robot programs. Among other things, this justifies a previous specification of the planning problem in the presence of sensing. This is joint work with Fangzhen Lin.

Description Logics and their Applications
Abstract only

Maurizio Lenzerini
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, I-00198 Roma, Italy
lenzerini@dis.uniroma1.it

Description Logics are logics for representing and reasoning about classes of objects and their relationships. They can be seen as successors of frame systems and semantic networks, and have been investigated for more than a decade from different points of view, in particular, expressive power and computational complexity of reasoning. In the talk, I first review the research done in the past years in Description Logics. Then I discuss the relationships with other formalisms, such as modal logics, database models, and object-oriented languages. Finally, I describe how Description Logics have been applied in several fields, including software engineering, configuration management, databases, and information systems.

Panel Abstract

PANEL
Themes at the Co-Located Workshops

Lin Padgham
RMIT
Department of Computer Science
GPO Box 2476V
Melbourne 3001, Australia
linpa@cs.rmit.edu.au

For KR'98 it was decided to attempt to co-locate meetings of a number of more specialised but established research communities, whose members were likely to also have an interest in KR. The Description Logics Workshop had been successfully co-located with KR'96 and there appeared to be benefits for all involved.

Co-locating a number of KR-related workshops is expected to facilitate interchange between specialised research communities with resulting cross-fertilisation of ideas. It is often at the interface between research communities that new and fruitful ideas arise.

The panel "Themes at the co-located workshops" will attempt to identify current trends within each specialist area that are related to the more general KR area, or that are common across a number of the specialist areas.

Co-locating with KR'98 are the following events:

- Nonmonotonic Reasoning Workshop (NMR)

  May 30 - June 1, 1998. Contact person: Gerhard Brewka (brewka@informatik.uni-leipzig.de), chairs: Gerhard Brewka and Ilkka Niemelae, honorary chair: Ray Reiter.

  The aim of the workshop is to bring together active researchers interested in the area of nonmonotonic reasoning to discuss current research, results, and problems of both a theoretical and practical nature. This time the workshop consists of plenary sessions and specialized workshops on the following topics:

  - Formal Aspects and Applications of Nonmonotonic Reasoning (Co-chairs: Jim Delgrande, Mirek Truszczynski)
  - Computational Aspects of Nonmonotonic Reasoning (Co-chairs: Ilkka Niemel, Torsten Schaub)
  - Logic Programming (Co-chairs: Juergen Dix, Jorge Lobo)
  - Action and Causality (Co-chairs: Vladimir Lifschitz, Hector Geffner)
  - Belief Revision (Co-chairs: Hans Rott, Mary-Anne Williams)

  A preliminary list of invited speakers includes Bob Kowalski, Isaac Levi, John McCarthy, Dan Roth and Erik Sandewall.

- Workshop on Validation & Verification of Knowledge-Based Systems (V&V'98)

  June 1, 1998. Contact person: Frank van Harmelen (frankh@cs.vu.nl).

  The European V&V community has traditionally met at workshops during the European AI Conferences (ECAI). The purpose of the change of venue to co-location with KR'98 is to encourage exchange of ideas between researchers in the field of knowledge-representation with those concentrating on V&V of KBS specifically. There are obvious connections between the two fields, and papers on topics focussing on this connection are especially solicited. related to

- Workshop on Knowledge Representation for Interactive Multimedia Systems (KRIMS II)

  June 1, 1998. Contact persons: George Vouros (georgev@ath.aegean.gr), Paolo Petta (paolo@ai.univie.ac.at).

  The target audience of KRIMS-II includes researchers in the areas of collaborative systems, personal assistants, intelligent multimedia presentation systems, adaptive interfaces, multimedia information retrieval, and intelligent integration of information.

  Major topics to be addressed include:

  - knowledge representation schemes for multimedia information repositories
  - supporting interactivity in content determination and/or communication of information.
  - knowledge representation for interactive multimedia information content determination, integration, co-ordination and presentation.
  - methodologies and systems for construction and maintenance of knowledge bases for interactive multimedia systems.
  - methodologies and systems for construction and maintenance of interactive multimedia DBMS.

- International Description Logics Workshop (DL'98)
  June 6 - June 8, 1998. Contact person: Enrico Franconi (franconi@irst.itc.it).

  The 1998 International Workshop on Description Logics (DL'98) continues the tradition of international workshops devoted to discussing developments and applications of knowledge representation formalisms based on Description Logics. Topics include Foundations of Description Logics, Extensions of Description Logics, Integration of Description Logics with other formalisms and Use of Description Logics in applications.

- Conference on Formal Ontology in Information Systems (FOIS'98)

  June 6 - June 8, 1998. Contact person: Nicola Guarino (guarino@ladseb.pd.cnr.it).

  Research on ontology is becoming increasingly widespread in the computer science community. Its importance has been recognized in fields as diverse as qualitative modelling of physical systems, natural language processing, knowledge engineering, information integration, database design, geographic information science, and intelligent information access. The conference will have a strongly interdisciplinary character. Expected participants include computer science practitioners as well as linguists, logicians, and philosophers. Although the primary focus of the conference is on theoretical issues, methodological proposals as well as concrete applications from a well-founded theoretical perspective will be discussed.

Most of these events represent an established group that have held previous well-attended meetings. Some communities have been meeting regularly over a large number of years, whereas others are relatively new, responding to newly arising research areas.

Having several meetings that one wants to attend located together makes it more likely that a researcher is able to attend both meetings. From the point of view of a workshop, organisers may well be able to attract new contributors to the research community as participants wish to attend KR. From the point of view of KR, more people may be able to arrange travel funding due to having a paper at a workshop, and will also attend KR. Co-locating the events above will hopefully facilitate attendance of a larger number of interested people and thus enrich all of the communities.

AUTHOR INDEX

336

PRINCIPLES OF KNOWLEDGE REPRESENTATION AND REASONING: PROCEEDINGS OF THE SIXTH INTERNATIONAL CONFERENCE

Edited by Anthony G. Cohn (University of Leeds, UK), Lenhart Schubert (University of Rochester, New York), and Stuart Shapiro (State University of New York at Buffalo)

The Knowledge Representation (KR) conferences have established themselves as the leading forum for timely, in-depth presentation of progress in the theory and principles underlying the representation and computational manipulation of knowledge.

The papers in this volume, which are twice as long as papers in general AI conferences, have passed a stringent review process. The topics covered include description, graph-based, and nonmonotonic logics; logic programming–based representations; spatio-temporal, probabilistic, modal, and contextual reasoning; theory building, merging, and revision; diagnosis, reasoning about actions, planning, and execution; and representing granularity and vagueness. Within these topics, treatments range from the abstract specification of algorithms and their computational complexity to the analysis of implemented systems.

These proceedings are an essential reference volume for researchers and students interested in a detailed view of this key research area.

ISBN 1-55860-554-1
ISSN 1046-9567
Artificial Intelligence

Morgan Kaufmann Publishers
340 Pine Street, 6th Floor
San Francisco, CA 94104
http://www.mkp.com

ISBN 1-55860-554-1
90000>
9 781558 605541